AN ITALIAN EVENT-BASED ASR-TTS SYSTEM FOR THE NAO ROBOT

Piero Cosi*, Giulio Paci*, Giacomo Sommavilla*, Fabio Tesser*
Marco Nalin**, Ilaria Baroni**
*Institute of Cognitive Sciences and Technologies, ISTC, C.N.R., UOS Padova, Italy
** Hospital San Raffaele, Milano, Italy
**[piero.cosi, giulio.paci, giacomo.sommavilla, fabio.tesser]@pd.istc.cnr.it*
***[nalin.marco, baroni.ilaria]@hsr.it*

1. ABSTRACT

This paper describes an event-based integration approach for building a human-robot spoken interaction system using the NAO robot platform with the URBI middleware within the ALIZ-E project. The ALIZ-E integrated system includes various components but we mainly concentrate on the Automatic Speech Recognition (ASR) and the Text To Speech (TTS) synthesis modules while the following Natural Language Understanding (NLU), Dialog Management (DM) and Natural Language Generation (NLG) ones will be only briefly introduced. We describe these components and how we adapted and extended them for use in the system. We discuss several options that we have considered for the implementation of the interfaces and the integration mechanism and present the event-based approach we have chosen. We describe its implementation using the URBI middleware. The system has been be used for HRI experiments with young Italian users since April 2011.

2. INTRODUCTION

Conversational systems play an important role in scenarios without a keyboard, e.g., when talking to a robot. Communication in human-robot interaction ultimately involves a combination of verbal and non-verbal (gesture) inputs and outputs. Systems capable of such an interaction thus must process verbal and non-verbal observations in parallel, as well as execute verbal and non-verbal actions accordingly in order to exhibit synchronized behaviors. The development of such systems involves the integration of potentially many components and ensuring a complex interaction and synchronization between them.

NAO (Aldebaran Robotics, 2012; Gouaillier et alii, 2008) is one of the most often used humanoid robots for academic purposes worldwide and it is complete with a user-friendly programming environment. From simple visual programming to elaborate embedded modules, the versatility of NAO and its programming environment enables users to explore a wide variety of subjects at whatever level of programming complexity and experience.

Figure 1: NAO, the ALIZ-E robot.

In this paper we present an event-based approach for integrating a conversational robotic system. This approach has been instantiated using the URBI middleware (URBI Open-Source, 2012; Baillie, 2005) on a NAO robot that is being used as a test bed for investigating child-robot interaction in the context of the ALIZ-E[1] project (ALIZ-E, 2012).

Within the ALIZ-E project the need for a robust software environment capable of performing ASR (Automatic Speech Recognition) and TTS (Text To Speech) synthesis for Italian children users interacting with NAO has been made manifest. In the ALIZ-E project, we choose alternatively SPHINX (CMU SPHINX, 2012; Lee et alii, 1990) or JULIUS (JULIUS, 2012; Lee and Kawahara, 2009) ASR and ACAPELA (ACAPELA, 2012) or MaryTTS (MARY TTS, 2012; Schröder, 2003) TTS, four complete state-of-the-art software suites that implement all the components needed for speech recognition and synthesis respectively. Also a VAD (Voice Activity Detection) module (Dekens and Verhelst, 2011) which triggers all speech processing activities on NAO has been integrated in the system. Moreover, URBI has been elected as the middleware that “orchestrates” the many components that form part of the ALIZ-E project architecture. URBI aims to be a universal, powerful and easy-to-use software for robotic programming. It is based on a client/server architecture providing a high-level interface for accessing the joints of the robot or its sensors.

Using the event-based approach to system integration introduced above, we have been developing a system in the ALIZ-E project integrating the following components: Audio Front-End (AFE), Voice Activity Detection (VAD), Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Dialogue Manager (DM), Natural Language Generation (NLG), Text-To-Speech Synthesis (TTS), Non-verbal Behavior Planning (NVBP) and motor Control (MC). The integration of the components is shown in Fig. 2: filled boxes indicate components implemented in Java, double-line boxes components in C/C++, and plain boxes components in URBIScript. The TTS component with the marble filled box is either the ACAPELA TTS on the NAO or the Mary TTS implemented in Java.

[1] ALIZ-E develops embodied cognitive robots for believable any-depth affective interactions with young users over an extended and possibly discontinuous period

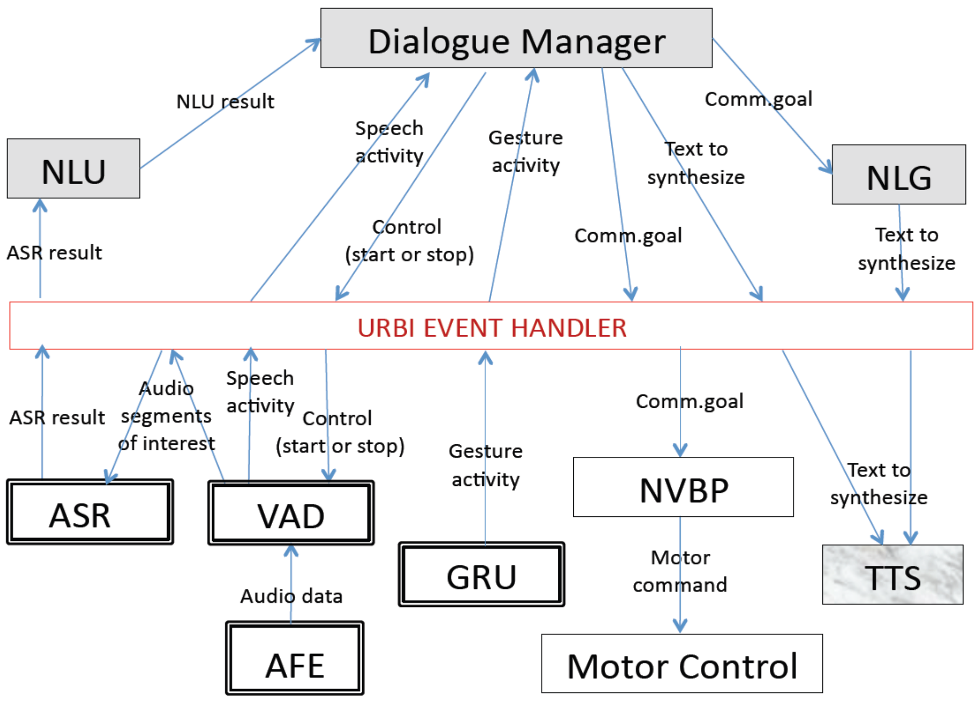


Figure 2: The components of the integrated system.

The system has been so far implemented for three scenarios: an imitation game of arm movements, a quiz game, and a dancing game and this paper focuses on the components developed for the imitation and quiz games, in which verbal interaction has a key role. The system for the dancing game applies the same integration approach. Experiments with Italian children in all three scenarios have been carried out in Milan at Hospital San Raffaele.

3. ASR

Two widely used open-source ASR systems have been considered for integration into the ALIZ-E framework: SPHINX and JULIUS.

SPHINX has been developed at Carnegie Mellon University (CMU) at Pittsburgh, while JULIUS has been developed at Kyoto University and both systems have to be considered as Large Vocabulary Continuous Speech Recognition (LVCSR) systems with real-time decoding capabilities.

Originally in the ALIZ-E project we used CMU SPHINX-3 (CMU SPHINX, 2012; Lee et alii, 1990) for speech recognition of Italian children users interacting with NAO but we are now using JULIUS because, in SPHINX-3, it was difficult to implement run-time features (live decoding) and to handle audio input. Moreover, SPHINX-3 is no longer maintained and, on the other hand, JULIUS has proven to be very easy to be integrated within the ALIZ-E project architecture.

3.1. JULIUS

Open-Source LVCSR Engine JULIUS is a high-performance ASR decoder for researchers and developers, designed for real-time decoding and modularity. Moreover most of the features available in other state-of-art decoders are also available for JULIUS, including major search techniques such as tree lexicon, N-gram factoring, cross-word context dependency handling, enveloped beam search, Gaussian pruning, Gaussian selection, etc.

JULIUS decoder main features are:

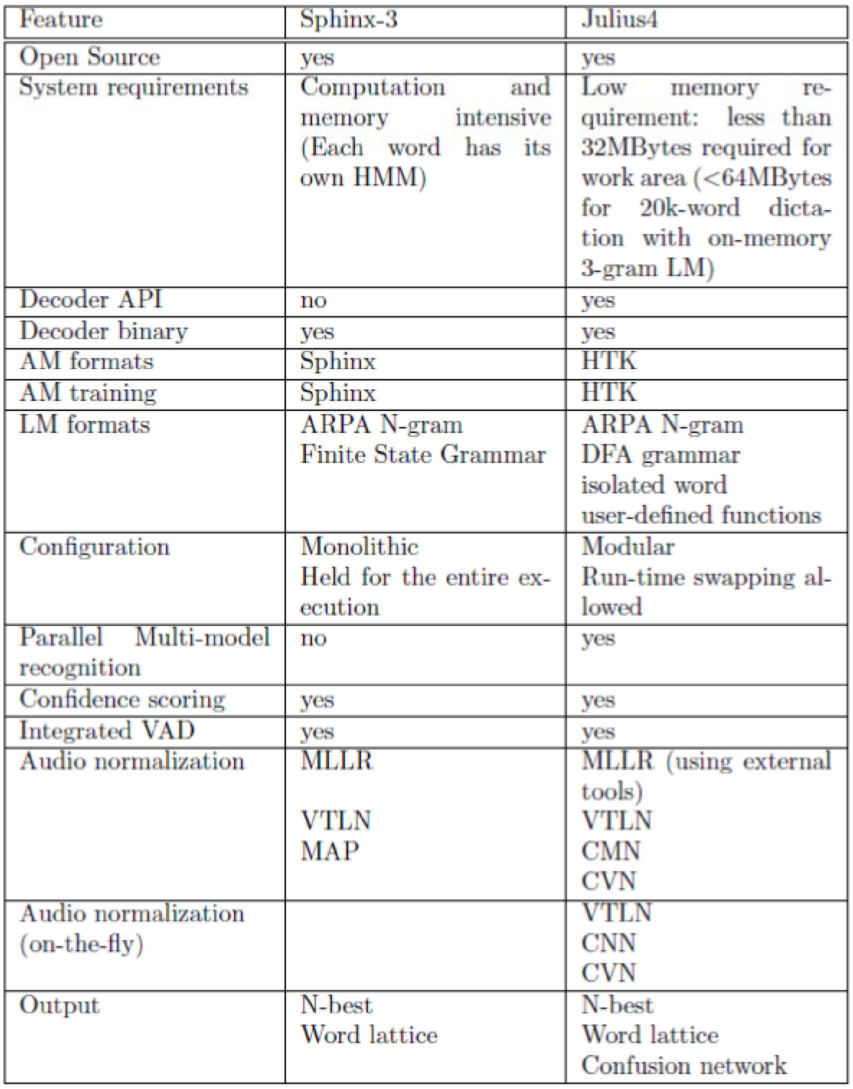
- open-source;
- small memory footprint;
- core engine is a separate C library;
- AM Models in HTK (HTK, 2012) ASCII hmmdefs format are supported;
- several LM types are supported: N-gram, grammar, isolated word and user-defined LM function embedding;
- modular configuration file structure (i.e., each config file can embed another one covering only one particular aspect of the configuration);
- it can perform decoding with parallel, multiple LMs and AMs: from the same audio input, a different result is given for every ac./lang. model;
- N-best / Word lattice / Confusion network output;
- word-level confidence scoring;
- real-time, on-the-fly, high speed, two-pass (forward-backward) decoding:
  - word bigrams are used for the first pass, while N-grams are used for the second one;
  - first pass output can be used before second one has finished;
- integrated GMM-based and Energy-based VAD;
- on-the-fly audio normalization: cepstral mean normalization (CMN), cepstral variance normalization (CVN), vocal tract length normalization (VTLN).

Most of these features are desirable in the ALIZ-E project.

3.2. SPHINX-3 and JULIUS-4 Comparison

In Table 1 a comparison between SPHINX-3 and JULIUS-4 is provided. The following elements in particular showed that JULIUS is more suitable than SPHINX-3 for the ALIZ-E project and drove the ASR engine change decision:

- JULIUS decoder API is very well designed (it made integration smoother in comparison with SPHINX-3);
- its system requirements are low (this is important in an integrated system handling several components);
- language models can be swapped at run-time;
- configuration is modular;
- possibility of performing multi-model recognition;
- on-the-fly Input/AM normalization.



Feature	Sphinx-3	Julius4
Open Source	yes	yes
System requirements	Computation and memory intensive (Each word has its own HMM)	Low memory requirement: less than 32MBytes required for work area (<64MBytes for 20k-word dictation with on-memory 3-gram LM)
Decoder API	no	yes
Decoder binary	yes	yes
AM formats	Sphinx	HTK
AM training	Sphinx	HTK
LM formats	ARPA N-gram Finite State Grammar	ARPA N-gram DFA grammar isolated word user-defined functions
Configuration	Monolithic Held for the entire execution	Modular Run-time swapping allowed
Parallel Multi-model recognition	no	yes
Confidence scoring	yes	yes
Integrated VAD	yes	yes
Audio normalization	MLLR VTLN MAP	MLLR (using external tools) VTLN CMN CVN
Audio normalization (on-the-fly)		VTLN CNN CVN
Output	N-best Word lattice	N-best Word lattice Confusion network

Table 1: SPHINX-3 and JULIUS-4 feature comparison.

3.3. Acoustic Model Training with HTK

The LVCSR Engine JULIUS distribution does not include specific training tools for acoustic models, however any tool that create acoustic models in the Hidden Markov Model Toolkit (HTK) format can be used. In the ALIZ-E Project we used the HTK tools (HTK, 2012; Young et alii, 2006) for this task, following the Voxforge HTK training for JULIUS tutorial (VoxForge, 2012b). The same procedure has been used to train an Italian adult acoustic model, using the training data provided for the EVALITA 2011 Forced Alignment task (Cutugno et alii, 2012) (a subset of the CLIPS corpus) and an Italian child acoustic model, using the FBK CHILD-IT corpus (Gerosa et alii, 2007). The training procedure is described in the following Table 2.

1. create a prompts file with the transcription of all audio files;
2. create a pronunciation dictionary that covers the whole prompts file (our own lexicon has been used; the lexicon has been extended with pronunciations of missing words using Sequitur G2P (Bisani and Ney, 2008), grapheme to phoneme tool followed by a quick manual revision);
3. obtain a list of all the used phones, including silence phones;
4. generate a phone level transcription from the orthographic transcriptions, using the first entry of the pronunciation dictionary;
5. extract 13 MFCC coefficients (plus Δ) from audio files (25ms Hamming analysis window, 10ms analysis step, preenphasis 0,97);
6. initialise model parameters for embedded training (13 normalized MFCC features plus Δ, 25ms Hamming analysis window, 10ms analysis step, preenphasis 0,97);
7. perform embedded training using HERest tool (Young et alii, 2006). Embedded training has the advantage that it does not require any prior knowledge of the phone boundaries and thus can be applied when only orthographic transcription is available (only symbolic phonetic transcription is required). HERest implementation of embedded training is implemented using the Baum-Welch algorithm. It starts by initialising to zero the accumulators for all the parameters of all the HMMs and then, for each training utterance, proceeds as follow:
   (a) joins all the HMMs of the phonetic symbols of the phonetic transcription, constructing a composite HMM;
   (b) calculates the forward and backward probabilities for the composite HMM;
   (c) calculates the probabilities of state occupation at each time frame using the forward and backward probabilities;
   (d) updates the accumulators in the Baum-Welch usual way;
   finally it uses the accumulators to estimate new parameters for all of the HMMs;
8. repeat the embedded training steps two times more;
9. add a short pause model by copying the learned silence model parameters and allowing short pauses between words;
10. repeat the embedded training steps two times more;
11. perform forced alignment using the current acoustic model, allowing multiple pronunciations, in order to detect the best one;
12. repeat the embedded training steps two times more using the new phonetic transcriptions;
13. convert the phonetic transcription (that uses context independent phones) to a context dependent one (using left and right phones as context);
14. repeat the embedded training steps two times more using the new phonetic transcriptions;
15. manually create a file with phones features derived from linguistic knowledge;
16. tie phone states according to phonetic questions and a lexicon of all the words to be recognised;
17. update the phonetic transcriptions according to the tied phone states;
18. repeat the embedded training steps two times more using the new phonetic transcriptions;

19. create a map for all the remaining possible phone states, so that words not in the training set nor in the recognition lexicon can be recognised in a later step.

Table 2. Acoustic Model Training Procedure.

3.4. Language modelling

The LVCSR Engine JULIUS supports N-gram, grammar and isolated word Language Models (LMs). Also user-defined functions can be implemented for recognition. However its distribution does not include any tool to create language models, with the exception of some scripts to convert a grammar written in a simple language into the Deterministic Finite Automaton (DFA) format needed by the engine. This means that external tools should be used to create a language model.

3.4.1 N-gram LM

The JULIUS engine supports N-gram LMs in ARPA format. We used SRI-LM toolkit (SRILM, 2012; Stolcke, 2002), to train a simple model for question recognition of the Quiz Game ALIZ-E scenario. The Quiz questions and answers database has been used as training material for this model. The model is very simple and very limited, but it should be enough to recognise properly read questions (the questions to be recognised are expected to be from the training set), especially if used in conjunction with some other, more flexible, model.

3.4.2 Grammar LM

The JULIUS engine distribution includes some tools that allow to express a Grammar in a simple format and then to convert to the DFA format needed by JULIUS. That format, however, has very few constructs that helps writing a proper grammar by hand and writing a non-trivial grammar is very hard. Third-party tools exist to convert an HTK standard lattice format (SLF) to the DFA format and to optimise the resulting DFA [8]. SLF is not suitable to write a grammar by hand, but HTK provides tools that allow a more convenient representation based on the extended Backus-Naur Form (EBNF) (Young et alii, 2006).

A simple model for Quiz answers recognition where written in the EBNF-based HTK grammar language. Part of the grammar was automatically derived by including the answers in the Quiz database. Several rules were added to handle common answers and filler words.

3.5 ASR Resources

In this section the resources used for Acoustic Model building and the work done for expanding them are described. Efforts have been made to create an ALIZ-E system test set, consisting of transcribed spontaneous speech (i.e., a corpus of speech plus transcription data recorded from ALIZ-E experiments).

3.5.1 .Speech Corpora

Two Italian and one English Speech Corpora have been tested so far with HTK and JULIUS:

- the training data provided for the EVALITA 2011 Forced Alignment task (Cutugno et alii, 2012) (this is a subset of the Italian CLIPS Corpus adult voices that counts about 5 hours of spontaneous speech, collected during map-task experiments, from 90 speakers from different Italian areas);
- Italian FBK ChildIt Corpus (Gerosa et alii, 2007) (this is a corpus of Italian children voice that counts almost 10 hours of speech from 171 children; each child reads

about 60 children literature sentences; the audio was sampled at 16 kHz, 16 bit linear, using a Shure SM10A head-worn mic);

- English Voxforge adult voices (Voxforge, 2012a) (the whole 16 kHz, 16 bit linear data set as it was in October 2011 has been used; the data set counts more than 80 hours of read speech from more than 600 speakers).

3.6 ASR Data collection

Both read and spontaneous speech has been collected for the ALIZ-E project. Read speech has been recorded according to guidelines similar to those that were used to collect the FBK ChildIt corpus, with the main goal of extending training material. Spontaneous speech has been collected during real interactions between children and NAO, with the main goal of creating a proper test set for further development.

3.6.1 Read Speech

Data collection of read speech is useful to enlarge our Corpus of audio plus transcription children data. The major advantage of collecting read speech is that obtaining the transcriptions corresponding to the audio is straightforward. Thus is a relatively low time consuming task (compared to transcribing spontaneous recordings). These data are meant to be used to train the Acoustic Model.

Several sessions has been recorded during a summer school near Padova. For the text of the recordings it has been decided to use the FBK Childit’s prompts which are phonetically balanced sentences selected from children literature. For each session the input coming from the four NAO microphones, a close talk microphone (Shure WH20QTR dynamic head set) and a panoramic (AKG Perception 200, -10 dB, flat equalisation) one has been recorded. The close talk and the panoramic microphones were connected to a digital audio recorder (Zoom H4n Handy Recorder). Synchronisation of the sources has been granted using a chirp-like sound, played by an external loudspeaker, at the beginning of every utterance.

The goal within this project is to collect read speech for a total amount of ~10 hours / ~10000 utt. / 171 children, which is the size of FBK ChildIt. About 2 hours of children speech (~2k utterances, 32 children) have already been collected and more will be collected in the future. NAO was very useful as its use helped to keep children attention alive.

3.6.2 Spontaneous Speech

Read Speech is very useful for expanding AM training data, but it is not well suited for building a reliable test set for ASR in the ALIZ-E project. A proper test set should consist of audio collected in a scenario as close as possible to the real one. For this reason it has been decided to manually transcribe and annotate some data collected during the Quiz game experiments that took place at Ospedale San Raffaele in December 2011 and March 2012.

The collected audio data consists of spontaneous speech recordings of children utterances produced during real interactions with NAO in a Wizard-Of-Oz modality. The database will be extended with data recorded from new experiments. The experiments consisted basically of a Q&A quiz game between the robot and the child. It starts with the robot asking questions (and subsequently providing possible answers) to the child, then, after more or less four answers, they exchange roles and the child became the asker. In the latter case, the child speech cannot be considered entirely “spontaneous”. Anyway, since it is part of the real system interaction, we consider that those data can represent a good test set.

Sometimes the child is not sure if robot heard him/her, and repeat his/her input, thus in those cases we collect more than one sample utterance of the same sentence. Also, we noticed a few cases of barge-ins: the user knew the answer and answered without waiting for the options. The robot gave them anyway, and the child re-answered. Frequent cases of interjections were observed from the children: the user does not understand robot's answer (and usually says "Uh?").

It has been decided not to use NAO built-in microphones for recording, as those are low quality microphones. Moreover, two of them are placed under the speakers, one is placed on the robot's nape, near the fan, and all the four microphones record a lot of noise resulting from motors and electronic circuits. Instead, we used hand-free radio close talk microphone (Proel RM300H, Radio frequency range: UHF High Band 750-865 MHz, Microphone: headset HCM-2). This microphone has been selected in order to interfere as little as possible with child-robot interaction. The microphone has been connected to a Zoom H4n, allowing us to record both using a computer (the Zoom is used as an USB audio input interface) or saving audio data on a SD memory card.

The first experiments showed that there were no problems in convincing the kids to wear the microphone. Telling them that "it helps NAO hearing you" is enough. These experiments also seems to confirm the impression that radio wireless technology can be really useful for (1) giving a reasonable freedom of movement to the users and (2) sending clean speech data to the system PC.

We started transcribing the recorded experiments' utterances with the software Transcriber (Transcriber, 2012; Barras et alii, 1998), version 1.5.1. The fundamental rule for transcribing was to use correct orthographic lowercase words with punctuation, except for the following specific cases:

Numbers: do not transcribe them as digits, but in orthographic form (for ex. if heard /VOGLIO LA 1/, then transcribe: "voglio la uno");

Acronyms: as they are spelled: they can be pronounced:

(1) as a word (for ex. /AIDIESSE/: "AIDS_aidiesse");
(2) as a combination of names of letters and a word (for ex. /GEIPEG/: "JPEG_geipeg")
(3) only as the names of letters (for ex. /AIDS/: "AIDS_aids");

Letters: (for ex. /A/: "A", /BI/: "B", /ZETA/: "Z", /B/: "B b", /C/ (as in "cane"): "C k", /C/ (as in "cielo"): C TS);

Proper Nouns: put initial capital letter (for ex. /MARIO/: "Mario");

Interruptions: insert an hyphen as suffix (for ex. /VADO A CA/: "vado a ca-"); be aware that, depending on the context, there can be differences (for ex. /CHE BELLO IL CE/ (meaning "cesto") transcribe "che bello il ce-", otherwise (meaning "cielo"): transcribe "che bello il cie-");

Wrongly pronounced words: transcribe in uppercase the correct word, fol¬lowed by an underscore and the wrongly spelt word, trying to adhere to heard sounds as much as possible with Italian orthography (for ex. /PROABILMENTE/: "PROBABILMENTE_ proabilmente");

Loanwords: transcribe with correct (foreign) orthography (for ex. /UIKEND/: "week-end" /OCHEI/: "okay");

Foreign words: here again, try to adhere to heard sounds as much as possible with Italian orthography (for ex. /GION/: "JOHN_gion" /GARAGE/: (with the french /Z/) "GARAGE_ garaZ");

Speaker fillers: here follows a list of the most frequent fillers we encountered so far: (a) <EHM>, (b) <mmm>, (c) <mmm>, (d) <EH>, (e) <whispered->, (f) <-whispered>, (g) <lipsmack>, (h) <breath>, (i) <tongueclick>, (j) <laughter>, (k) <unintelligible>;

External noises: here follows a list of the most frequent noises we encountered so far: (a) <robot speaks>, (b) <robot moves>, (c) <garbage>, (d) <background voices>.

3.7 .JULIUS/URBI integration into the ALIZ-E system

One of the main problems in a robotic environment is how to deal with the need of having components that either:

(a) can access to low-level hardware details:
(b) perform heavy computations, and typically run on different, more powerful machines:
(c) should be coordinated concurrently:
(d) should react to (typically asynchronous) events.

Languages such as C/C++ can well suit low-level and heavy computational tasks, but it can be tedious to manage concurrency, network communication and event handling with those programming languages.

The open source URBI (URBI, 2012; Baillie, 2005) environment, a Universal Robotic Body Interface, developed by GOSTAI ALIZ-E project partner, provides a standard robot programming environment made of urbiscript scripting language which can orchestrate complex organisations of components named UObjects in highly concurrent settings.. It is based on a client/server architecture where the server is running on the robot and accessed by the client. The URBI language is a scripted language used by the client and capable of controlling the joints of the robot or access its sensors, camera, speakers or any accessible part of the machine.

It is relatively easy to make a C/C++/Java program accessible to the URBI language. The UObject API defines the UObject class and each instance of a derived class in C++ code will correspond to an URBI object sharing some of its methods and attributes. Basically one needs to "wrap" the upstream program into a C++ (or Java) class inheriting from Urbi::UObject, then bind in urbiscript the methods that should be accessible from there. Eventually, one needs to define events and how to handle methods concurrently.

3.7.1 URBI ASR component

In the ALIZ-E integrated system, ASR is provided as an API whose functions can be accessed by other components (e.g., the DM - Dialog Manager or the NLU - Natural Language Understanding module). When ASR output is available an event is launched and the result is provided as a payload, so that every components that needs this information can access it. ASR component is basically made up of two modules: a configuration structure (that holds data for AM and LM) and a main recognition loop function (also called "JULIUS stream"). The latter contains an internal VAD - Voice Activity Detection module and outputs second pass recognition result as an NBest list.

The principal methods of this component are: load/free/switch configuration; start/stop main recognition loop. A schema of function call and data exchange among ASR, DM and NLU components can be seen in Figure 3.

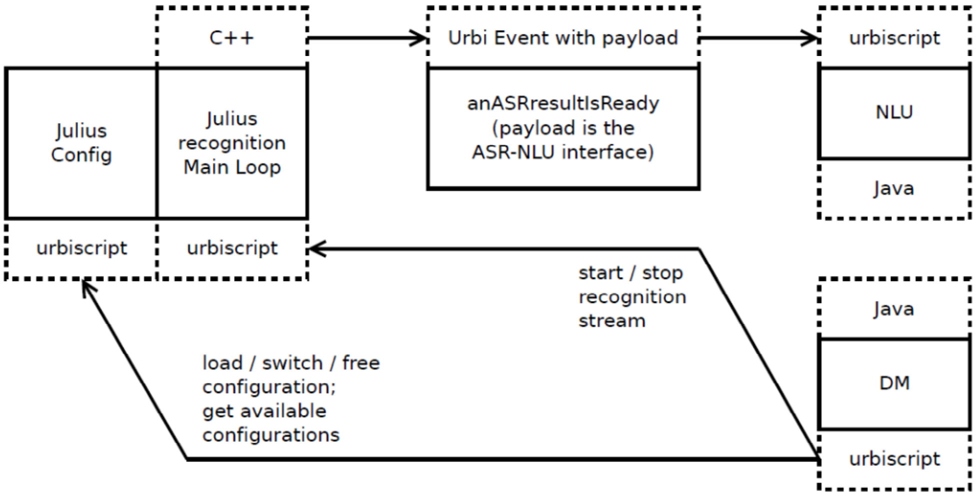

C++
Julius Config
Julius recognition Main Loop
urbiscript
urbiscript
Urbi Event with payload
anASRresultIsReady (payload is the ASR-NLU interface)
urbiscript
NLU
Java
start / stop recognition stream
load / switch / free configuration; get available configurations
Java
DM
urbiscript

Figura 3. ASR communication through urbiscript.

3.8 ASR-NLU Interface

In order to communicate with other modules such as the NLU Java one, we needed to create an URBI data structure that could be populated by JULIUUS C++ UObject output, and that could be accessed by other components.

3.8.1 Data Exchange Format

Thanks to the collaboration with DFKI project partner, responsible for the ALIZ-E Natural Language Understanding and Dialogue Management tools, we implemented an NBest interface for feeding the NLU component with ASR output. This Data Structure consists of a list of sentences; each of them holds total (i.e. sentence-wide) acoustic and linguistic probabilities. Sentences are actually lists of words, and for every word an acoustic probability and a linguistic one are provided.

It is worth noting that Sphinx-3 actually provides by default acoustic and linguistic probabilities for both word- and sentence-level, while JULIUS outputs sentence-level acoustic and linguistic probabilities, and only a generic confidence score for word-level.

3.8.2 Communication Reliability

In order to implement this data exchange interface in the ALIZ-E integrated system we had to solve 3 major problems:

1. in the ALIZ-E integrated system, components communicate each other over the network, thus we need a way to encode the structured ASR output into something that can be sent through a serial communication channel, and then decode it (i.e. “restore” the data structure) on the receiver side;

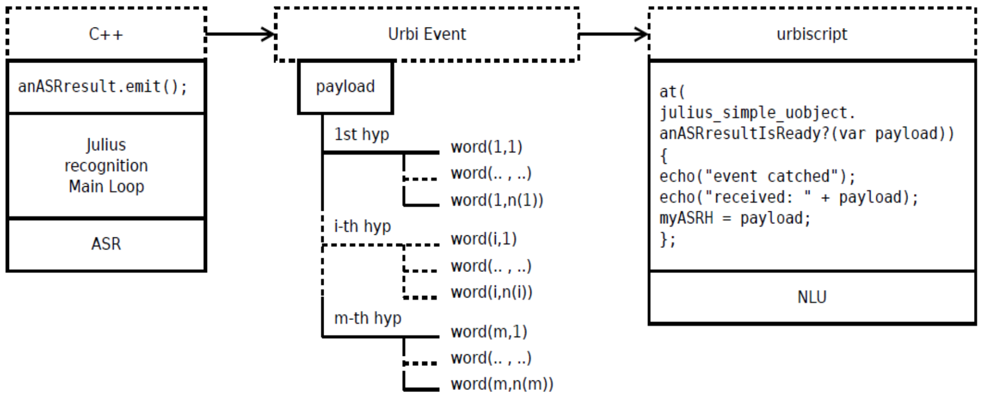
2. suppose that the ASR component uses a shared variable to store its computation result and alerts other components which in turn will have to read the content. In this case is not possible to guarantee that the latter are able to do so before the variable is overwritten.
3. to overcome the above problem, one may consider using a complex structure (like a queue) to save successive outputs; in this way, however, there is the problem of emptying this data structure, as it may be difficult to know whether all the components that were to receive the recognition result have read the contents of the variable.

The URBI middleware layer provides features that address these problems. Problem 1 is solved by serializing ASR output data structure. Regarding Problems 2 and 3, we safely implemented this interface thanks to the event-based programming paradigm. With URBI, setting triggering events and reacting to them is straightforward; also, the following conditions are guaranteed:

- messages are passed one-to-many (multicast or broadcast);
- messages are transferred reliably (multicast protocol reliability means that the system ensures total order, atomicity and virtual synchrony);
- messages are guaranteed to be delivered in order;
- messages are passed asynchronously: the sender delivers a message to the receivers, without waiting for them to be ready (this is well suited in robotic environments where messages are generated at irregular intervals).

In our current implementation the result of ASR is sent to NLU component, but, since URBI allows multicast signaling, events can easily be caught by more than one component in the integrated system.

Actually, the C++ JULIUS UObject sends (or “emits”) an “ASRresultIsReady” URBI event, which is caught by urbiscript code that triggers NLU functions. The ASR output Data Structure (NBest list) is embedded to the event as a “payload”. This means that the event carries a message for the receiving components, ensuring atomicity in the communication. Figure 4 shows a diagram in which the ASR component emits an event with payload that is caught by NLU.



```
at(
julius_simple_uobject.
anASRresultIsReady?(var payload))
{
echo("event catched");
echo("received: " + payload);
myASRH = payload;
};
```

Figure 4. ASR to NLU event with payload.

4. TTS

Within the ALIZ-E project, the robot have to convey different messages to the child. Movements/gestures, lights and audio/speech are the different output channels available on the NAO robot (Gouaillier et alii, 2008) and they are used to communicate the desired message in this project. With reference to the voice channel only, it is known that a lot of messages are contained in the speech signal. Table 3 shows the main messages and the related speech correlates. These messages are encoded by particular acoustic patterns recognizable in the human speech and it would be attractive for a speech synthesizer, to be able to synthesize these patterns in order to communicate to the child all these different messages using the audio channel.

Message	Speech correlates
speaker identity	spectral envelope, voice quality
emotional state of the speaker	spectral envelope, voice quality, prosody
verbal content of the speech	spectral envelope
syntactic information	spectral envelope, prosody
focus	prosody

Table 3. Messages contained in the speech signal and their acoustic correlates. Here the term focus refers to the part of a sentence which expresses the centre of attention.

Moreover emotional speech synthesis must take into account the manipulation of paraverbal parameters like speech rate, voice intensity, pause durations, etc.

4.1 HMM or Statistical Parametric Synthesis and its use on a robotic system

The Statistical Parametric Synthesis (or HMM Speech Synthesis) (Zen et alii, 2009) approach has been chosen for the task of modeling the voice of the robot, because it allows to act on the produced acoustic patterns in various ways and it seems the most suitable solution allowing stronger parameter control than Unit Selection synthesis. For example using HMM speech synthesis technology it is possible to:

- change the speaker identity of the synthetic voice; this is possible changing the vocoder (Imai, 1983; Fukada et alii, 1992) parameters or alternatively using speaker adaptation techniques (Yamagishi et alii, 2009);
- stress the focus of a sentence; applying some particular prosodic patterns;
- change the emotional content of the synthetic speech applying different prosodic settings and patterns;

The prosodic settings and patterns mentioned beforehand can be either the results of previously acquired knowledge and experience (e.g., it is knows that happy voices usually adopt an higher pitch with respect to a normal voice) or they can be the results of modules able to learning these from real data.

Some TTS customizations have been already implemented and used in the ALIZ-E project, the first is used to provide the robot with a child-like voice, the others refer to the possibility to apply prosodic modifications according to the focus or to produce a speech that reflect a particular emotional state of the robot.

Finally, in order to obtain emotive speech, HMM trajectory estimation techniques must be coupled with digital signal processing algorithms and speech models capable of implementing voice quality and timbre modifications (Tesser et alii, 2010) as well as general pitch shifting and time stretching algorithms. In fact, while it is true that HMM-based speech synthesis allows for more flexible voice control, data-driven speech synthesis allows for more natural sounding voice qualities.

4.2 Robot voice identity

Synthesized speech triggers social identification processes, for this reason within the ALIZ-E Project we would like to use the voice of a child for NAO. A vocal tract scaler, which can simulate a longer or shorter vocal tract, has been used in order to obtain a child-like voice, starting from a female voice.

In this implementation the frequency axis warping method has been used. The resulting voice is good for this task because the robot doesn't need to have a realistic child voice and some artifacts can be accepted.

Anyway, using the HMM synthesis approach, some improvements in this task are possible, for example:

- to use a vocal tract scaler effect based on vocoder used in system (MLSA filter (Imai, 1983; Fukada et alii, 1992) with mixed excitation (Yoshimura et alii, 2001);
- to use speaker adaptation techniques for HMM synthesis (Yamagishi et alii, 2009)

4.2 Focus prominence by prosodic modification

The Natural Language Generation module is able to mark focus words during the verbal output generation process. In the speech synthesis process we are interested in emphasizing the focus using adequate speech parameters. As HMM synthesis technology is suited to prosody modifications, a first implementation has been done using some tags able to force the relative prosody changes in the words that bears the focus. After some informal listening test, the prosody on the focus words are forced in the following way:

- the speech rate is decreased of 10% with respect to the normal production;
- the pitch is raised of 25% with respect to the normal production.

4.3 Emotional prosodic modification

The ALIZ-E system is able to decide when the verbal output should be rendered with (non-neutral) emotional coloring, either "sadly" or "happily".

According to this, the speech paralinguistic feedback is realized increasing the speech rate (+5%) and the pitch contour (+25%) in the happy case, while in the sad case the speech rate and pitch contour are both decreased (-20%).

4.3 Italian voice for Mary TTS

MaryTTS (MARY TTS, 2012; Schröder, 2003) is an open-source, multilingual Text-to-Speech Synthesis platform written in Java. It was originally developed as a collaborative project of DFKI's Language Technology lab and the Institute of Phonetics at Saarland University and is now being maintained by DFKI. As of version 4.3, MaryTTS supports German, British and American English, Telugu, Turkish, and Russian and more languages are in preparation. MaryTTS comes with toolkits for quickly adding support for new languages and for building unit selection and HMM-based synthesis voices.

Within the ALIZ-E project we have pursued the route of making available an Italian MaryTTS female voice for the robot. We started with the porting of some of the existing Italian FESTIVAL TTS modules (Cosi et alii, 2001, Tesser et alii, 2005). An Italian lexicon for MaryTTS has been created converting the Italian FESTIVAL lexicon and an Italian Letter-To-Sound (LTS) module has been obtained together with a first simple Part Of Speech (POS) tagger.

The latest official version of MaryTTS (4.3.1) contains the support for Italian and the istc-lucia-hsmm voice. It can be downloaded from http://mary. opendfki.de/wiki/4.3.1.

4.4 Text corpus selection

In order to select the text scripts for the recordings we have used the automatic procedure for optimal (phonetically/prosodic balanced) text selection made available in MaryTTS (Pammi et alii, 2005), based on the analysis of the freely available Wikipedia dumps for the language taken into consideration.

The original MaryTTS procedure has been modified to select only sentences for whose it was possible to obtain a phonetic transcription using only the lexicon. The final text selection has been obtained by the iteration (4 times) of the following steps:

- ignore all sentences that do not improve the coverage score;
- manual inspection of the selected list and removal of the too-difficult-to-pronounce sentences;
- reiterate the coverage selection procedure.

Table 4 shows the technical details of the obtained text corpus.

Feature	Description
Wikipedia dump date	2011/08/15 (2011081519411313430062)
DB Size (sent.)	1400
Coverage method	SimpleDiphones+SimpleProsody

Table 4. Description of the Italian Text corpus for Mary TTS.

4.5 Recordings

In order to do not fatigue the speaker's voice, the recordings has been done in several sessions spanned in two weeks. Table 5 shows the technical details of the recordings.

Feature	Description
Speaker	Female
Age	20
Room characteristics	Silent room
Microphone	Shure WH20QTR Dynamic Headset
O.S.	Linux Ubuntu
Soundcard	Focusrite Saffire LE FireWire audio interface
Audio driver	Jack Sound Server trough Pulse Audio
DB Size (sentences)	1400
DB Size (time)	~2 hours
Manually checked segmentation	Only on sentences identified by a quality control check

Table 5. Description of the Lucia TTS recording corpus.

4.6 Building the voices

Both Unit Selection and HMM voices have been created using the Voice Import Tools under the MaryTTS environment. The building process consists of the following steps:

- feature extraction from acoustic data;
- feature vector extraction from text data;
- automatic labeling (ehmm from Festvox);
- Unit Selection voice building;
- HMM voice building (SPTK, 2012; HTS, 2012).

The resulting voices was positively judged by some informal listening test with the following comments:

- the Unit Selection voice has good audio quality, but sometimes the voice is cracked/chunked, probably because of some missing units in the corpus;
- the HMM voice has a lower audio quality, but it has an higher intelligibility and constant quality with respect to the Unit Selection voice.

4.7 TTS Integration into the ALIZ-E system

With regards to integration we wanted to keep the possibility of using both TTS systems: ACAPELATTS (ACAPELA, 2012) and MaryTTS (MaryTTS, 2012). We achieve this implementing a configuration system able to choose between the two different TTS system to use.

Since ACAPELATTS is already available on NAO, most of the work has been done for making available MaryTTS into the NAO/URBI environment.

MaryTTS is a platform written in Java using the client/server paradigm. Due to the NAO CPU resources limitations, it has been decided to run the MaryTTS server on the remote PC while keeping the ACAPELA TTS as a built-in system on the NAO Robot.

When MaryTTS produces the audio stream the resulting speech must be played on the NAO loudspeaker. This has been achieved using a streaming server based on GStreamer (GStreamer, 2012). In order to have a real time interaction, a Real-time Transport Protocol (RTP) (RTP-Wikipedia, 2012) streaming server is active on NAO. URBI is able to manage RTP data as well, but we have decided to follow the GStreamer approach in order to avoid URBI server overloading. Moreover this approach allows to choose among a lot of already available plug-in for audio/video pipelines building.

In order to bring MaryTTS and GStreamer RTP in the URBI world, an URBI Object (UMaryTTS) has been created as the principal Object responsible for routing the synthesis request (Mary TTS client) and for playing the resulting audio to different output channels.

These channels are represented by the following Urbi Objects:

- UMaryTTSAudioPlayer is an UObject that makes a request to the MaryTTS server and play the resulting synthesized voice through the PC loudspeakers (useful for the fake robot simulation);
- UMaryTTSAudioPlayer is an UObject that makes a request to the MaryTTS server and streams the resulting synthesized audio through an RTP connection[2] using the efflux library (efHux-java, 2012);

[2] This connection is created and destroyed every time a sentence is synthesized.

- UMaryTTSGstreamerPlayer is an UObject that makes a request to the MaryTTS server and stream the resulting synthesized audio through an UDP RTP permanent connection[3] using GStreamer-java (GStreamer-java, 2012).

While ACAPELA TTS exposes the functions: say(pText) and stopTalking(), UMaryTTS also exposes sayWithEmotion (pText, pEmotion) that is able to change the global prosody setting according to the emotion taken into consideration. Moreover UMaryTTS is able to manage the focus[4] generated by the NLG module.

7. FUTURE PLAN

As for ASR, since mismatches between training and test conditions can severely degrade the performance, one can apply Speaker Adaptation to ASR, using a small amount of observed data from an individual speaker to improve a speaker-independent model.

Adaptation can be very useful in the ALIZ-E project because children's vocal tract lengths can differ a lot. As the child will interact several times with the robot we can consider of reusing data from previous interactions for adapting the models. Preliminary experiments using SPHINX have shown recognition improvements on ChildIt corpus using for example VTLN (Vocal Tract Length Normalization) and MLLR (Maximum Likelihood Linear Regression) adaptation techniques (Nicolao and Cosi, 2011).

ASR test set such audio recordings have to meet very precise requirements to be considered reliable for our purposes:

- they should be collected from children (as much spontaneous as possible) speech;
- they must reflect a real user case project scenario.

Due to the these reasons, we have not yet been able to evaluate ASR and NLU accuracy because we lack of test data. Nonetheless, this is a major goal for our purposes. We are currently involved in speech data collection in various ALIZ-E Experiments conducted at HSR.

An AI Component based on the Dialogue or the Memory Manager of the ALIZ-E integrated system could be conceived in order to "choose" a specific LM according to its current status. Such specific models should be available before the interaction starts.

For example, if the interaction is in a preliminary stage, the AI component could prefer to load a LM that is built upon greetings and introductory speech forms. Fig. 5 shows a schema of this idea. Also LM online generation which is related to the previous idea cold be exploited assuming that AI components in the system can have access to text data (collected from children real speech) and can build LMs at runtime from specific subsets of those data using AI knowledge, the interaction status and the history (memory).

In order to improve JULIUS recognition performance we aim to collect more data for the ChildIt2 corpus, audio plus transcription, which will be used to train a better, more robust children AM for Italian.

[3] We experienced that a permanent RTP connection is more stable than a single shgot conncetion for each sentence, then UMaryTTSGstreamerPlayer is more stable than UMaryTTSAudioPlayer.

[4] The special characters § and ^ are treated as symbols to identify the focus part in a sentence.

The input of our JULIUS component is currently fed by the capture device of ALSA, and we want to implement a connection between the input of the recognizer and the Audio Front End (AFE) module. This will bring more flexibility to the system, and the possibility of testing the four NAO built-in microphones.

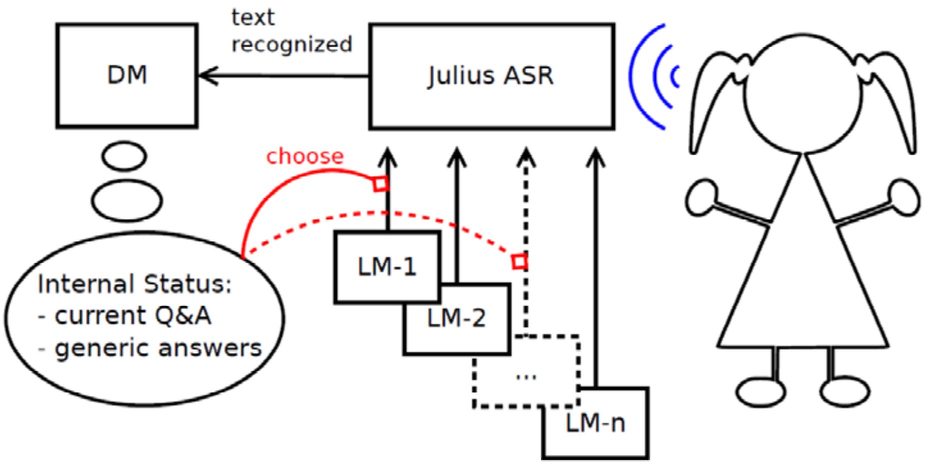


Figure 5. Run-Time LM switching schematic representation.

AM/voice adaptation could be used in our system to improve recognition accuracy. One idea could be to adapt the AM with a portion of speech pronounced at the beginning of a interaction. The adapted AM could be used for the rest of the dialogue.

The recognition engine currently works with a fixed pronunciation dictionary and we want to build a letter-to-sound module for Italian, in order to phonetize (and thus be able to recognize) new words and we are considering to test two l2s engines for the Italian language (trained upon our database):

- MaryTTS letter2sound;
- Sequitur;

As for TTS, in order to improve the current use of speech synthesis inside the ALIZ-E project, the main effort will be spent on improving the coupling between the Natural Language Generation (NLG) module and the speech generation using a more sophisticated input for the TTS. We will test some ideas about the use of automatic prominence detection (Rosenberg, 2010) inside the HMM voice building creation process; this can results in a method that allows to force some prosodic symbols like ToBI (Silverman et alii, 1992) together with text in input to the TTS.

Besides, we would like to investigate strategies to achieve a better child-like voice for the robot inside the ALIZ-E project. For example we plan to evaluate the building of a MLSA-based vocal tract effect, and testing the HMM speaker adaptation/voice conversion algorithms.

Regarding the Italian MaryTTS system, the following improvements will be evaluated:

- to build a more robust POS tagging module for Italian;
- to build a text normalization module able to handle digits, acronyms, etc.;
- to re-check the phonetic prosodic coverage of the corpus;

- to re-run the text scripts selection by using these more evolved Natural Language features;
- if necessary other recordings with the same speaker will be taken into account.

6. CONLUDING REMARKS

We introduced an event-based integration approach for building a human-robot spoken interaction system using the NAO robot platform with the URBI middleware within the ALIZ-E Project. In particular, we focused on how we adapted and extended the JULIUS Large Vocabulary Continuous Speech Recognition (LVCSR) system and the MaryTTS Text To Speech (TTS) synthesis modules. Their final integration into the system is a first important mark toward the implementation of a fully integrated communication system for NAO.

We discussed several options considered for the implementation of the interfaces and the integration mechanism and presented the event-based approach we have chosen. We described its implementation using the URBI middleware.

The system has been be used with success for HRI experiments with young Italian users since April 2011.

ACKNOWLEDGMENTS

Parts of the research reported on in this paper were performed in the context of the EU-FP7 project ALIZ-E (ICT-248116).

INDEX TERMS: Human-Robot Interaction (HRI), integration, NAO, URBI, Italian children Automatic Speech Recognition (ASR) , Italian Text-To-Speech (TTS) synthesis, Voice Activity Detection (VAD), Dialogue Management (DM), Natural Language Generation (NLG), Non-Verbal Behavior Generation (NVBG).

REFERENCES

ACAPELA (2012), URL: http://www.ACAPELA-group.com/index.html.

Aldebaran Robotics (2012), URL:: http://www.aldebaran-robotics.com/en.

ALIZ-E (2012), URL: http://ALIZ-E.org/.

Baillie, J. C. (2005), "URBI: Towards a Universal Robotic Low-Level Programming Language", in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, 820-825.

Barras, C., Geoffrois, E., Wu, Z., and Liberman, M. (1998), Transcriber: A Free Tool for Segmenting, Labeling and Transcribing Speech, Proc. Of the First International Conference on Language Resources & Evaluation (LREC), Granada, Spain, 1998, 1373-1376,

Bisani, M., and Ney, H. (2008), Joint-sequence models for grapheme¬to-phoneme conversion, Speech Communication 50.5 (2008), 434-451.

CMU SPHINX (2012), URL: http://cmuSPHINX.sourceforge.net/.

Cosi, P. and Nicolao, M. (2009), "Connected Digits Recognition Task: ISTC-CNR Comparison of Open Source Tools", in CD Proceedings of EVALITA Workshop 2009, in Poster and Workshop CD Proceedings of the XI Conference of the Italian Association for Artificial Intelligence, 2009, Reggio Emilia, Italy, December 9-12.

Cosi, P., Tesser, F., Gretter, R. and Avesani, C. (2001), “Festival Speaks Italian!”, in Proceedings of Eurospeech’01, , 2001, Aalborg, Denmark, September 3-7, 509-512.

Cutugno, F., Origlia, A. and Seppi, D. (2012), EVALITA 2011: Forced alignment task. Tech. rep. 2012. URL: http://www.evalita.it/sites/evalita.fbk.eu/files/working_notes2011/Forced_Alignment/FORCED_ORGANIZERS.pdf.

Dekens, T. and Verhelst, W. (2011), “On the Noise Robustness of Voice Activity Detection Algorithms”, Interspeech 2011, Florence, Italy, 2649-2652.

efflux development team (2012), efflux: a Java RTP Stack with RTCP Support and a Clean API, Mar. 2012, URL: https://github.com/brunodecarvalho/efflux.

Fukada T., Tokuda, K., Kobayashi, T., and Imai, S.(1992), An Adaptive Algorithm for Mel-Cepstral Analysis of Speech, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1992, Vol. 92-1, 137–140.

Gerosa, M., Giuliani, D. and Brugnara, F. (2007), “Acoustic variability and automatic recognition of children's speech. Speech Communication, 49 (Feb 2007), 847-860.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J. and Maisonnier, B. (2008), “The NAO Humanoid: a Combination of Performance and Affordability”, CoRR, 2008, abs/0807.3223 (http://arxiv.org/abs/0807.3223).

GStreamer Development Team (2012), GStreamer Open Source Multimedia Framework, Mar. 2012, URL: http://gstreamer.freedesktop.org/.

GStreamer-java Development Team (2012), Java Interface to the GStreamer Frame-Work, Mar. 2012, URL: http://code.google.com/p/gstreamer-java/.

HTS Working Group (2012), HMM-Based Speech Synthesis System (HTS) version 3.2. Mar. 2012, URL: http://hts.sp.nitech.ac.jp/.

HTK - Hidden Markov Model Toolkit (2012), URL: http://htk.eng.cam.ac.uk/

Imai, S., (1983), Cepstral Analysis Synthesis on the Mel Frequency Scale, Proc. IEEE ICASSP 1983, Vol. 8., 93-96.

JULIUS (2012), URL: http://JULIUS.sourceforge.jp/en_index.php.

Lee, A. and Kawahara, T. (2009), "Recent Development of Open-Source Speech Recognition Engine JULIUS", Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2009.

Lee, A., Kawahara, T. and Shikano, T., (2001), Julius - an Open Source Real-Time Large Vocabulary Speech Recognition Engine, Proc Interspeech 2001, 1691-1694.

Lee, K.F., Hon, H. W. and Reddy, R. (1990), “An Overview of the SPHINX Speech Recognition System”, IEEE Transactions on Acoustics, Speech and Signal Processing, January 1990, 38(1): 35-45.

Mary TTS Development Team (2012) The MARY Text-to-Speech System, Mar. 2012, URL: http://mary.dfki.de/.

Nicolao, M. and Cosi, P. (2011), “Comparing SPHINX vs. SONIC Italian Children Speech Recognition Systems”, in Abstract Book & CD-Rom Proceedings of AISV 2011, 7th Conference of Associazione Italiana di Scienze della Voce, “Contesto comunicativo e variabilità nella produzione e percezione della lingua”, Lecce, Italy, 2011 - Abs: 85 - (CD: 414-425).

Pammi. S., Charfuelan, M., and Schröder, M. (2005), Multilingual Voice Creation Toolkit for the MARY TTS Platform, Proc. Int. Conf. Language Resources and Evaluation, LREC 2005, URL: http://www.lrec-conf.org/proceedings/lrec2010/pdf/720_Paper.pdf.

Rosenberg, A. (2010), AuToBI - A Tool for Automatic ToBI annotation, Proc. Interspeech 2010, 146–149.

RTP-Wikipedia (2012), Real-time Transport Protocol. Mar. 2012. URL: http://en . wikipedia.org/wiki/Real-time_Transport_Protocol/.

Schröder M. and Trouvain, J. (2003), “The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching”, International Journal of Speech Technology, 6, 2003, 365-377.

Silverman, K., Beckman, M. Pitrelli, J., Ostendorf, M., Wightman, C., Price, P. Pierrehumbert, J., Hirschberg, J. (1992), TOBI: A Standard for Labeling English Prosody, in Ohala, J.J. et al. (eds.), Proceedings ICSLP 92, 867-870.
URL: http://www.cs.columbia.edu/\~{}julia/papers/TOBI_i92_0867.pdf.

SPTK Working Group (2012), Speech Signal Processing Toolkit (SPTK) version 3.2, Mar. 2012, URL: http://www.sp-tk.sourceforge.net/.

SRILM - The SRI Language Modeling Toolkit. URL: http://www.speech.sri.com/projects/srilm/.

Stolcke, A. (2002), SRILM - An Extensible Language Modeling Toolkit, Proc. Intl. Conf. on Spoken Language Processing, Denver, USA, vol. 2, pp. 901-904.

Tesser, F., Cosi, P., Drioli, C. and Tisato, G. (2005), “Emotional Festival-Mbrola TTS Synthesis”, in Proceedings of Interspeech’05, 2005, Lisbon, Portugal, 505-508.

Tesser, F., Zovato, E., Nicolao, M. and Cosi, P. (2010), “Two Vocoder Techniques for Neutral to Emotional Timbre Conversion, in Sagisaka, Y. and Tokuda, K., eds., Proceedings of 7th Speech Synthesis ISCA Workshop (SSW), Kyoto, Japan, 2010, 130–135.

Transcriber: a Tool for Segmenting, Labeling and Transcribing Speech (2012), Authors: Boudahmane, K., Manta, M., Antoine, F., Galliano, S., and Barras C. (2012), URL: http://trans.sourceforge.net.

URBI Open-Source (2012), URL: http://www.gostai.com/products/URBI/.

Yamagishi, J., Nose, T., Zen, H., Ling, Z., Toda, T., Tokuda, K., King, S., and Renals S., (2009), A Robust Speaker-Adaptive HMM-based Text-to-Speech Synthesis, IEEE Trans. Audio, Speech, & Language Processing, vol.17, no.6, 2009, 1208-1230.

Yoshimura, T.. “Mixed Excitation for HMM-based Speech Synthesis”. In: Eurospeech. 2001.

Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., and Kitamura, T. (2001), Mixed Excitation for HMM-Based Speech Synthesis, Prpc. EUROSPEECH 2001, 2263-2266.

Young, S.J, Evermann, G., Gales, M.J.F., Kershaw, D., Liux, X., Moore, G., Odell, J., Ollason, D., Povey, D., Valchev, V. and Woodland, P. (2006), The HTK Book, version 3.4. Cambridge, UK: Cambridge University Engineering Department, 2006.

Zen, H., Tokuda, K., and Black, A. W. (2009), “Statistical parametric speech synthesis”, Speech Communication, 51(11), 2009, 1039-1064.

VoxForge (2012a), Tutorial: Create Acoustic Model - Manually. Mar. 2012. URL: http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial.

VoxForge (2012b), Free Speech Recognition (Linux, Windows and Mac) - vox-forge.org. Mar. 2012. URL: http://www.voxforge.org/.