

Command Speech Interface to Virtual Reality Applications

Miloš Cerňak¹, Adrian Sannier

Department of Telecommunication
Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, SK
mcernak@ktl.elf.stuba.sk

Virtual Reality Applications Center
Iowa State University
4 Howe Hall, Ames, IA 50011, USA
sannier@iastate.edu

Technical Report
Iowa State University

Abstract

During last five years several attempts to develop the speech interface to especially simulation applications emerged due to the recent improvements in speech and language technology and the complexity of those application's interfaces. We describe our approach to control Virtual Reality applications via voice and GUI, in creation of simple multimodal command speech interface based on dialog modeling. For speech synthesis we used the Festival speech synthesis system, and are developed a sample of a new high-quality synthetic voice for the Air Traffic Control speech domain. We are proposing the architecture of Adaptive Limited Domain Speech Synthesis as an improvement of current Festival's Limited Domain Speech Synthesis. For dialog modeling and speech recognition we used the CSLU Toolkit, using its dialog layout tools to create a finite state diagram that models the supported dialog. Within CSLU Toolkit we implemented the grammar of commands by means of regular grammar formalism of PROFER (Predictive, Robust, Finite-state parser). Further we are proposing one PC test solution based on Windows platform for testing and evaluation of our speech interface.

Keywords: speech synthesis, speech recognition, dialog modeling, natural language processing, virtual reality

1 Overview

Command Speech Interface (CSI) at Virtual Reality Applications Center (VRAC) was implemented for improving the usability of control Virtual Reality (VR) applications inside of C6 Virtual Reality System (C6 is an immersive

¹ The work of this author was performed while visiting VRAC of IPTTR at Iowa State

environment for science and engineering challenges). It can be used simultaneously with GUI running on hand-held PC and so they together create a multimodal interactive interface to VR applications. We tested CSI with Bio Molecule application in C6 (see chapter 2.2) and with simple Cube application in a test environment (see chapter 4).

CSI supports commanding, confirming and questioning mode. In an Ex. 1 example, where “U” indicates user command and “S” system response, you can observe all modes:

Ex. 1: Communication modes

U Cube, change the color to red
S The color was changed
U What is the color?
S Red
U Tell me the color!
S Red

Multimodal interaction can be achieved by simultaneous using VR facilities and our CSI. An Ex. 2 shows multimodal interaction within Bio Molecule application:

Ex. 2: Multimodal interaction in C6

U MoleBio
S Yes
U <Targeting the atom 512 by means of VR facility>
U Go there!
S OK (or Atom number 512)

CSI can be used simultaneously for more VR applications. In this case, target application must be specified explicitly (e.g. by assigning the name for each application, like “Cubes” or “MoleBio”).

2 Command Speech Interface

The system consists of three main blocks:

- Speech Synthesis in Festival speech synthesis system²

² Festival is a general multi-lingual speech synthesis system, source:
<http://www.cstr.ed.ac.uk/projects/festival/>

- Dialog Modeling, Speech Recognition and Nature Language Processing in CSLU Toolkit³ (further only toolkit)
- Communication Bridge to VR application

The architecture of CSI is shown on Fig. 1. The toolkit uses a Tk/Tcl as interpreter language and therefore we investigated how to make a communication channel to VR application. At VRAC there was implemented distributed environment using CORBA, we decided to use some TCL extension as CORBA client. There was suitable Combat project⁴, but unfortunately it supported Tcl from version 8.1 and the toolkit supports only 8.0. Further attempt was done by

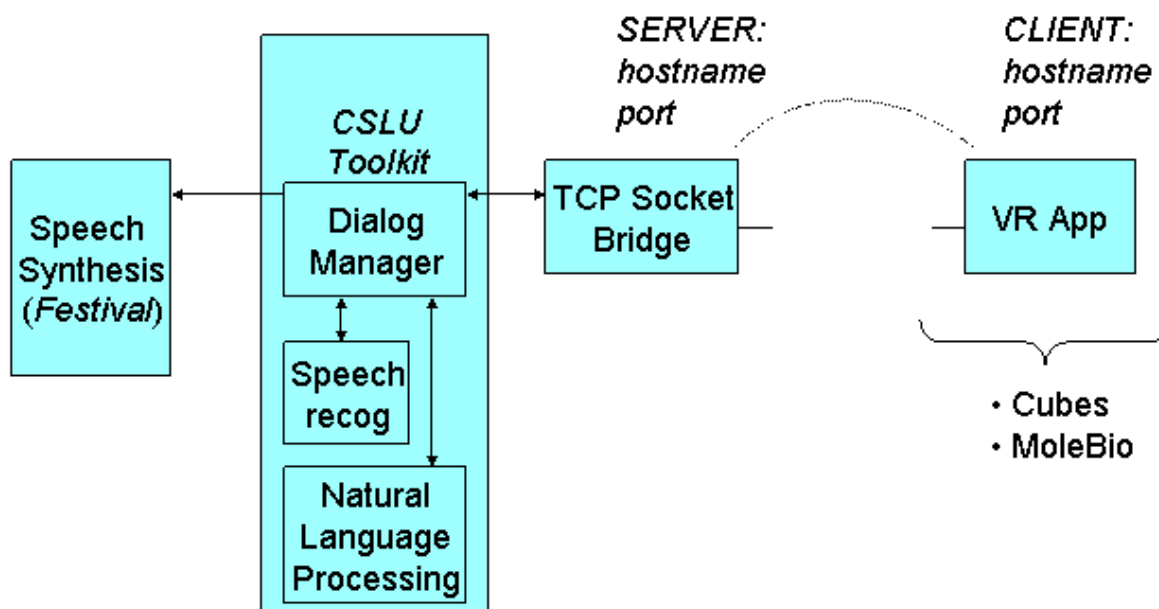


Fig. 1: Architecture of Command Speech Interface

using Tcl Blend⁵ what enables invoking JAVA methods and so JAVA client, who should call CORBA interface of VR application. Similar approach was used by GUI to control VP App. Here we encountered with a necessity to load into memory space of CSI whole JAVA client, because running JAVA client as a simple invoking of a JAVA method caused it was running in its own address space, and so the necessity to write another interproces communication. It was not effective and

³ CSLU Toolkit is a comprehensive suite of tools to enable exploration, learning, and research into speech and human-computer interaction, source: <http://cslu.cse.ogi.edu/toolkit/index.html>

⁴ Combat is a Tcl extension that allows accessing and providing CORBA objects at the script level, source: <http://www.fpx.de/Combat/>

⁵ Tcl Blend is a package for Tcl 8.x that allows you to load and interact with the Java VM, source: <http://tcl.activestate.com/software/java/>

therefore we chose direct TCP network communication between instance of the toolkit and VR app. The easiest way was to create a TCP server within the toolkit and add TCP client code to VR app.

One of the most crucial staff in the toolkit was to have some trigger to start speech recognition (it is depicted on Fig. 2). Rather than having some external device we used a feature “Leading Silence Duration” in the toolkit, where recording after that time is started again. This produces a quasi-continuous recording. For the trigger of the command recognition we further used a keyword followed by the command (see Ex. 1, the keyword is “Cubes”). At the end of the Leading Silence Duration time recording restart can be occurred while the user is preparing to say the first word. Generally this user’s preparation takes roughly 250 ms, so the probability of command’s cancellation P_{cc} by the toolkit is

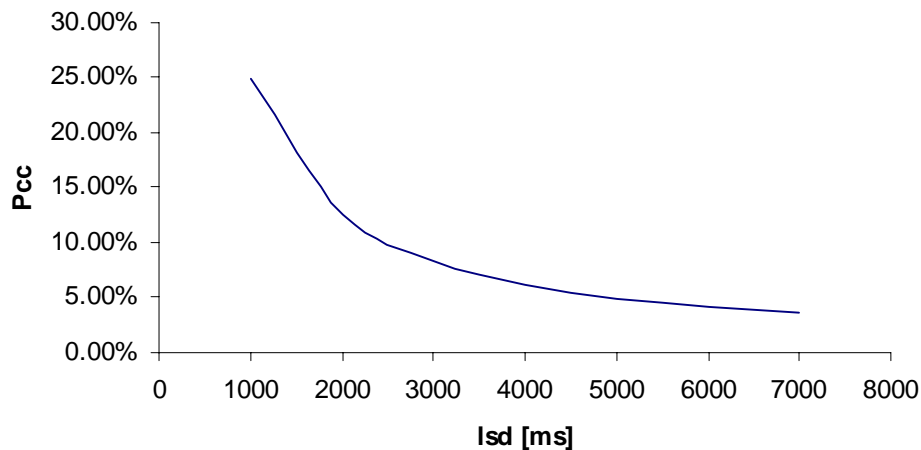
$$P_{cc} = 250 [ms] * 100 / lsd [ms],$$

where

P_{cc}	command cancellation probability [%]
lsd	Leading Silence Duration [ms]

The graph G. 1 shows downward progress of P_{cc} with upward lsd .

G. 1: Command Cancellation Probability



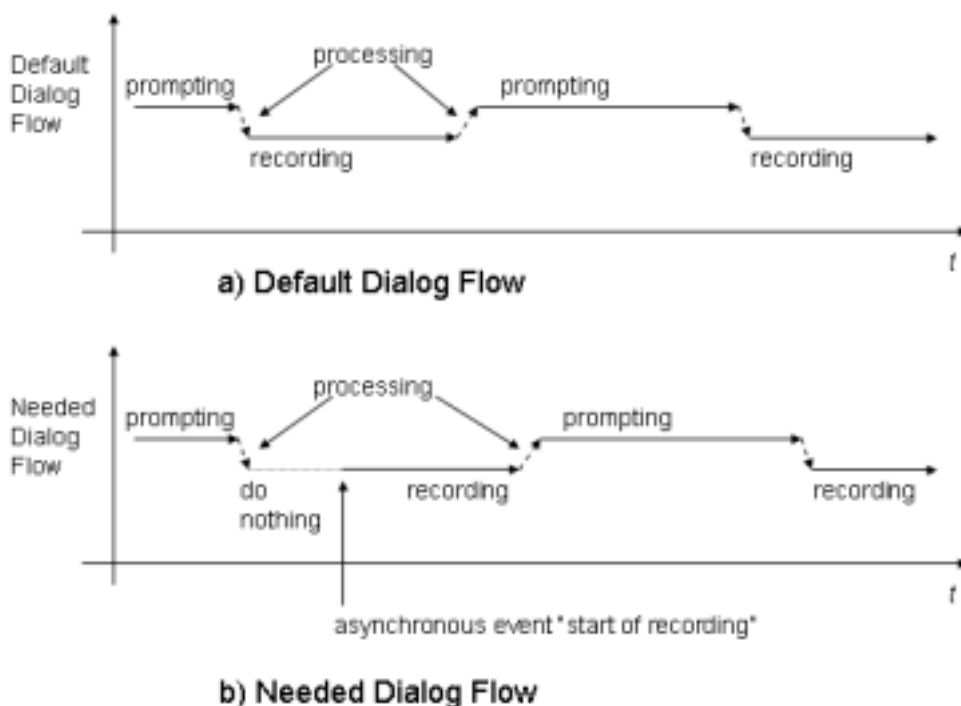


Fig. 2: Speech recognition starting

2.1 Natural language processing

For Natural Language Processing (NLP) we used PROFER within the toolkit. It allows regular grammar formalism for defining the pattern (e.g. command) that it will parse from the input [5]. We used its canonical parsing option, where the parser recognized output was mapped directly to TCP commands, which were being sent though open TCL channel. The most important stage is a design of the grammar text file, which would be further automatically compiled to usable grammar files during initialization of CSI application. Ex. 4 shows an example of such grammar file for cubes application. It supports commanding (equivalent commands 'color' and 'cubes') questioning mode (equivalent commands 'what' and 'tell me'). Only string ended with '_c' (_c stands for canonical) appears at the output. The grammar in Ex. 4 supports only tree colors, but it can be easily extended. Following rules in [5], more complicated grammar can be written. The grammar used in our tests use a vocabulary of about 40 words.

Ex. 4: Cubes command grammar

```
[cubes_query_type]
  ([command] [color])
```

```
[command]
  (color)
  (cube)
  ([question_c])
```

```
[question_c]
  (what)
  (tell me)
```

```
[color]
  ([red_c])
  ([blue_c])
  ([green_c])
  ([color_c])
```

```
[red_c]
  (red)
```

```
[blue_c]
  (blue)
```

```
[green_c]
  (green)
```

```
[color_c]
  (color)
```

2.2 Running in C6

We used our CSI also in real VR environment, in VRAC's C6. Audio signals were transmitted via wireless microphone. Due to computation expensive speech recognition evaluation, where are more possible hypotheses at the output, the computer running the toolkit had to be at least Pentium II, 400 MHz. Otherwise the delay of the toolkit brought unacceptable interruptions to the control of VR application.

We used CSI with moleBio application. Ex. 3 shows possible dialog flow. For command addressing we used a famous keyword "HAL".

Ex. 3: CSI demonstration in C6

U HAL
S Yes
U Color chain Domain five to purple
S OK
...
U HAL
S Yes
U Now turn on the ball number 1240 to green
S OK

Simultaneous to this command, the user in the cube could use hand-held PC and through JAVA client could also control the application. Ex. 2 shows such multimodal interaction.

3 Speech Synthesis

The simplest way to generate speech is its playing from the pre-recorded waveforms. These pre-recorded forms are very popular in the majority of IVR (Interactive Voice Response) systems. However, it is not an efficient way, if the number of prompts is very high (memory constraints) or the content is changing very often (repeating of time consuming recording of prompts). Therefore Text-to-speech (TTS) systems are successfully used for the speech generation, which nowadays mature to high-quality, intelligibility and their ability to be used within more complex systems, just like our CSI.

Our work has been begun with limited domain speech synthesis [1], where high-quality voice can be developed within a short time. We developed a sample of voice for the Air Traffic Control domain during a few days and used it from the toolkit. The voice was developed under Linux and then ported to Windows. The approach we have chosen is a concatenative speech synthesis, based on an offline clustering for unit selection speech synthesis [2] with detailed description of the process [3]. Due to the achievement of higher speech quality and the computational complexity of the method, the chosen unit is a phoneme together with the word it comes from. So the phoneme “a” from word “barnyard” (e.g. unit type a_barnyard) can be used to synthesize only word “barnyard”. This invokes the synthesis which works on the word concatenation basis, but it is not true – the synthesis is doing on the phoneme concatenation level. The question is, how could be synthesized out of

domain words. Each limited domain voice has also a back-off diphone voice for such situation. However, experiments showed [1] that switching the voice quality midway in a sentence is extremely distracting. Therefore if a phrase contains an out of vocabulary word, the whole phrase will be back-off. To solve this problem, we are proposing an Adaptive Limited Domain approach, described in following chapter 3.1.

3.1 Adaptive Limited Domain (ALD) Speech Synthesis

The ALD synthesis can be used within speech-in/ speech-out systems, where task report of dialog system [4] is used. The method is based on taking the content of missing word from dialog history, if the word was spoken before by user, and by means of voice conversion process the word is adapted to speech synthesis voice. Afterwards this new word is dynamically added to speech synthesis database, where new clusters are created or merely updated. The architecture is depicted in the Fig. 3.

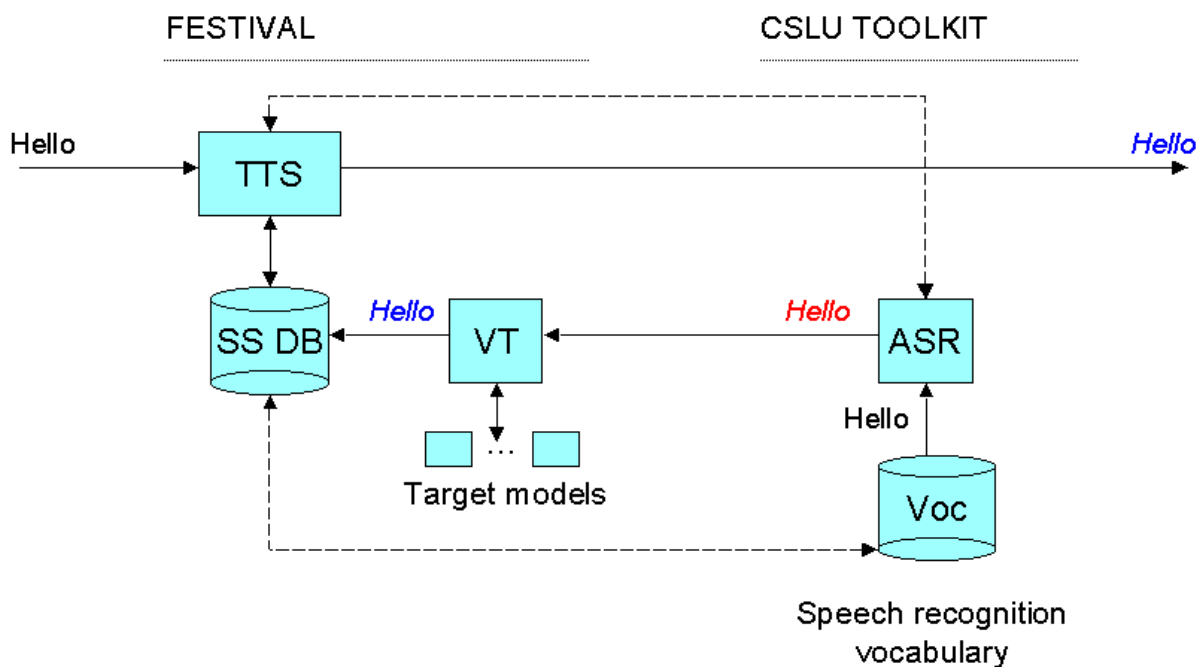


Fig. 3: Architecture of ALD Speech Synthesis

The task of the synthesis consists of the following processes

- Find recorded word within dialog manager from an Automatic Speech Recognition (ASR) block
- Voice transformation selected word
- Dynamic database modification
- Limited domain speech synthesis performing

It is necessary to back-up all recognized words together with assigned labels for later searching purposes. The search for particular word is done in label files. Once we have our missing out of vocabulary word, it is transformed to speech synthesis voice by OGI's voice conversion tool in Festival. Because voice transformation function must be trained for source and target voices, it must be done before using ALD synthesis. Relevant CSLU label file is than transformed to Festival format and all necessary processes within Festival must be run (extraction of pitchmarks and pitch-synchronous parameters) automatically. Finally, a cluster unit selection synthesizer is modified.

The process described is now still under construction and lots of question have to be solved (e.g. how to train voice conversion function, how to modify cluster synthesizer effectively, or how to search for out of vocabulary word within dialog manager).

4 SCI Test environment

For CSI evaluation we developed simple one PC test environment shown in Fig. 4, based on Windows platform. Fig. 4 shows VR application Cubes and the dialog is shown in Ex. 1.

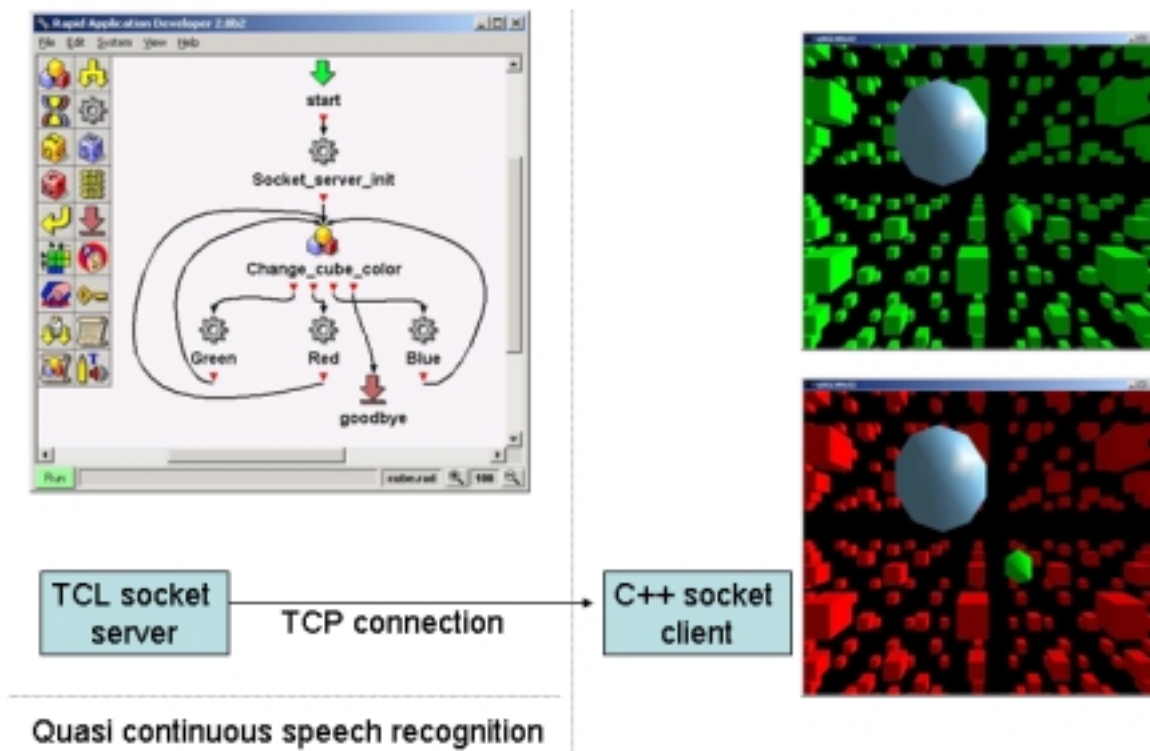


Fig. 4: SCI test solution

All VR applications are based on VR Juggler, what is a virtual platform for VR application development. Vr juggler, the toolkit and Festival are together installed and used within one PC. Of course, it also can be a distributed system with fore mentioned parts.

4.1 Evaluation of SCI

The main advantage of this system is quick deployment, very user friendly RAD (rapid application development) of the toolkit and high quality speech synthesis. It allows implementation of a speech-out/ speech-in system, with powerful extensibility to control another processes via Tcl script language. We were appreciated its fast dialog layout modification with immediate affects to VR applications. All stuff can be maintained by a small team, moreover an adaptation to new VR application is also quick enough.

The problematic area we encountered during our tests was speech recognition of the toolkit. Recognition accuracy was not sufficient and also tuning recognition parameters were more complicated than we expected. The only possibility to improve that is in own speech recognition training, but to do that for each new VR application domain is ineffective.

5 Related Work

The system is most closely related to Natural Language Interface to VR Systems [6], developed to facilitate control of 3D model of a portion of the interior a decommissioned US Navy ship. We improved an interaction with VR application in the terms of [6] by using a dialog modeling.

Probably the most sophisticated command interface, *CommandTalk* [7, 8], was developed by SRI International's Artificial Intelligence Center as a part of DARPA Communicator program⁶. The main difference is that *CommandTalk* uses an Open Agent Architecture (OAA) [11], likewise similar project *QuickSet* [9]. QuickSet is more oriented to multimodal interaction and defining multimodal integration strategy. *TRIPS*, an integrated AI (Artificial Intelligence) system [10], has also distributed architecture with KQML messages. Comparable to *TRIPS*, CSI has improved input modality (user interfaces), but poorer implementation of specified reasoners. The advantage of OAA is in its distributed philosophy and it acts as an integrator within a speech interface. Our CSI has distributed architecture only in terms of speech synthesis block, communication bridge block (TCP server can run on the other computer) and the toolkit (dialog modeling, NLP and speech recognition), but with careful design similar results could be achieved.

6 Future Work

The speech interface development lies at the top of spoken language technology implementation. Due to its complex, multi-disciplinary traits, usually a lot of people must work closely together. Speech technology in VRAC has begun with IBM ViaVoice implementation, has been continuing with current CSI and is going towards much usable, reliable and robust speech interface. The traits we have just mentioned are also future directions of CSI.

For an improving of the usability, the CSI compilation into one executable file must be enabled. Current version requires the toolkit loading, the project opening and finally the program running – this process seems to be ineffective. To achieve

⁶ The mission of the DARPA Communicator program is to develop a radically new capability within the information technology discipline that allows people to talk with computers, source: <http://www.darpa.mil/ipto/research/com/index.html>

this, it requires the manipulation with CSLU toolkit source. For reliability improving, speech recognition must be improved (see chapter 4.1). Finally, the robustness in terms of dialog possibilities must be added. The guidance from [8] can be taken, which mean correction, required argument inquiry, delayed response, consecutive correction and focusing features should be implemented within the dialog manager.

Nevertheless, parallel investigation of either OAA using [11] or a DARPA communicator program participation could be taken into account. In this case, the project should use Galaxy Communicator infrastructure software [12]. As first the Festival for speech synthesis and IBM ViaVoice for speech recognition should be chosen. Their cooperation can be facilitated by some JAVA program integrator. JAVA is very suitable for that task, because the Festival supports JAVA and also JSAPI (JAVA Speech API) from SUN. IBM ViaVoice also uses this JSAPI and JAVA integrator itself can contain JAVA client to connect CORBA ORB (Object Request Broker) and so a connection to Virtual Reality distributed facilities will be created. The other objects must be written, especially at first a dialog management object for (a) managing outgoing conversation (b) interpreting user communication in context and (c) selecting communicative actions to perform in response [10].

Implementation of proposed ALD Speech Synthesis using the toolkit and Festival frameworks is another work that should be done. At first a voice conversion within Festival has to be used to check a voice quality and then automatic cluster synthesizer modification has to be developed.

7 Acknowledgements

We would like to thank Ronald Sidharta for the cooperation with his moleBio application and to Institute for Physical Research and Technology at Iowa State University to enable the collaboration of the authors, under the aegis of Boeing Scholars Program.

References:

- [1] A. Black and K. Lenzo, *Limited Domain Synthesis*, Proceedings of the 6th Int. Conf. on Spoken Language Processing, Beijing, China, vol. II, p411-414, 2000.
- [2] A. Black and K. Lenzo, *Building Voices in Festival Speech Synthesis System*, pp 73-83, source <http://www.festvox.org>, 2000.
- [3] A. Black and P. Taylor, *Automatically clustering similar units for unit selection in speech synthesis*, Proceedings of Eurospeech 97, Rhodes, Greece, vol. II, pp 601-604, 1997.
- [4] McTear M., *Spoken Dialogue Technology: Enabling the Conversational User Interface*, Speech Technology Expert e-Zine Issue 5, pp 6-7, source <http://epublications.vercomnet.com/Speech/issue05/>, 2001.
- [5] Ed Kaiser, Michael Johnston, Peter A. Heeman, *PROFER: Predictive, Robust Finite-state Parsing for Spoken Language*, Proceedings of ICASSP, vol. II, pgs. 629-32, March, 1999.
- [6] Stephanie S. Everett, Kenneth Wauchope, Manuel A. Pérez-Quinones, *Future Fusion: Application realities for the virtual age*, Proceedings of VSMM98, 4th International Conference on Virtual Systems and Multimedia, Vol. II, pp. 469-474. IOS Press, Burke, VA, 1998; enhanced version on <http://www.aic.nrl.navy.mil/~severett/VSMM98/VSMM98.html>
- [7] Robert Moore et al., *CommandTalk: A Spoken-Language Interface for Battlefield Simulations*, Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, DC, pp. 1-7, Association for Computational Linguistics, 1997.
- [8] Amanda Stent et al., *The CommandTalk Spoken Dialogue System*, 1999, Proceedings of the 37th Annual Meeting of the ACL, pp. 183-190, University of Maryland, College Park, MD, Association for Computational Linguistics
- [9] Cohen, P. R et al. *QuickSet: Multimodal interaction for distributed applications*, Proceedings of the Fifth International Multimedia Conference (Multimedia '97), ACM Press, pp 31-40, 1997.

- [10] G. Ferguson and J. Allen, *TRIPS: An Intelligent Integrated Problem-Solving Assistant*, Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAI-98), Madison, WI., 1998.
- [11] Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S. C. *An open agent architecture*, in AAAI Spring Symposium, pp. 1-8, Mar. 1994. Reprinted in *Readings in Agents*, Huhns, M. and Singh, M. (eds), Morgan Kaufmann Publishers, San Francisco, 1998.
- [12] Bayer, S., Doran, Ch., George B., *Dialogue Interaction with DARPA Communicator Infrastructure: The Development of useful software*, in Proceedings of HLT 2001, First International Conference on Human Language Technology Research, J. Allan, ed., Morgan Kaufmann, San Francisco, 2001.