Instructions for running the ES 202 050 software

Five LINUX and five SunOS executable files are provided, the function of each of these programs is explained below. These are followed by example sets command lines for performing particular functions.

AdvFrontEnd *input_file.wav processed.mfc* -F RAW -fs 8 -q -swap
This program processes the speech data to produce the front-end features.

Inputs: Any raw speech file from an Aurora database (must be specified, e.g. *input.wa*v).

Outputs: File in HTK format, containing 14 dimensional feature vectors (must be specified, e.g. *processed.mf*c,).
Ascii file containing VAD flags for each frame (created automatically, *processed.vad*,).

Switches: -F RAW, input file format (NIST,HTK,RAW), -fs 8, sampling frequency in kHz (8 ,16), -swap, change input byte ordering (default is ieee-be), i.e. the swap switch will have to be set as appropriate, depending on platform and database.

coder_VAD *processed.mfc bitstream quantised.mfc* -freq 8 -Test_VAD
This program quantises the front-end features.

Inputs: As output by AdvFrontEnd. HTK format feature file vectors (must be specified, e.g. *processed.mf*c).
Ascii file containing VAD flags for each frame (read automatically, *processed.vad*,).

Outputs: Binary file for transmission that is discarded during ordinary testing but is required for error mitigation testing (must be specified, e.g. *bitstream*).
File in HTK format, containing quantised 14 dimensional feature vectors (must be specified, e.g. *quantised.mfc* or *dummy*), this file is discarded for error mitigation testing.
Ascii file containing VAD flags for each frame (created automatically when Test_VAD switch is set, *quantised.vad*,).

Switches: -freq 8, sampling frequency in kHz (8 ,16), -Test_VAD (reads input VAD file and creates an output VAD file) or –VAD (reads the input VAD file and outputs VAD flag in binary bitstream).

decoder_VAD *bit.out quantised.mfc* –VAD
This program decodes the binary bitstream and applies error mitigation. It is only required for the error mitigation testing.

Inputs: As output by coder_VAD.
Binary file, with or without channel errors (must be specified, e.g. *bitstream* or *bit.out*).

Outputs: File in HTK format, containing quantised 14 dimensional feature vectors (must be specified, e.g. *quantised.mfc).*
Ascii file containing VAD flags for each frame (created automatically when -VAD switch is set, *quantised.vad*,).

Switches: –VAD, reads VAD flag information from binary file and creates an output VAD file).

derivCalc *quantised.mfc final_output.mfc quantised.vad* -FD –COMB
This program carries out the server side post processing.

Inputs: As output by coder_VAD or decoder_VAD.
File in HTK format, containing quantised 14 dimensional feature vectors (must be specified, e.g. *quantised.mf*c)
Ascii file containing VAD flags for each frame vectors (must be specified, e.g. *quantised.va*d)

Outputs: Final HTK format file that is passed to the recogniser (must be specified, e.g. *final_output.mf*c).

Switches: -FD –COMB, these flags are compulsory.

resample_8_11 *11k_input_file 8k_input_file*
This program converts an 11kHz file to an 8kHz file (the file has to have no header)

Inputs: Any 11kHz raw speech file (must be specified, e.g. *11k_input_fil*e).

Outputs: An 8kHz raw speech file (must be specified, e.g. *8k_input_fil*e).

For General Testing at 8kHz:

AdvFrontEnd *input_file.wav processed.mfc* -F RAW -fs 8 -q -swap
coder_VAD *processed.mfc bitstream quantised.mfc* -freq 8 -Test_VAD
derivCalc *quantised.mfc final_output.mfc quantised.vad* -FD -COMB

For General Testing at 16kHz:

AdvFrontEnd *input_file.wav processed.mfc* -F RAW -fs 16 -q -swap
coder_VAD *processed.mfc bitstream quantised.mfc* -freq 16 -Test_VAD
derivCalc *quantised.mfc final_output.mfc quantised.vad* -FD -COMB

For General Testing at 11kHz:

resample_8_11 *11k_input_file 8k_input_file*
AdvFrontEnd *8k_input_file processed.mfc* -F RAW -fs 8 -q -swap
coder_VAD *processed.mfc bitstream quantised.mfc* -freq 8 -Test_VAD
derivCalc *quantised.mfc final_output.mfc quantised.vad* -FD -COMB

For Error Mitigation Testing at 8kHz:

AdvFrontEnd *input_file.wav processed.mfc* -F RAW -fs 8 -q -swap
coder_VAD *processed.mfc bitstream dummy* -freq 8 -VAD
"Error injection into bitstream" *bitstream bit.out*
decoder_VAD *bit.out quantised.mfc* -VAD
derivCalc *quantised.mfc final_output.mfc quantised.vad* -FD -COMB

where "Error injection into bitstream" represents whatever script is used to inject the EP1, EP2 and EP3 channel errors into the bitstream file in order to produce the errored bit.out file.

Test files in Test_Data

There are a 8kHz and a 16kHz raw speech file in the directory Test_Data (*v10770c7.it1.08* and *v10770c7.it1.16* respectively*)* as well as Linux and SunOS reference files in the Reference_Files subdirectory for each stage of processing. The 8kHz file is in big endian format (swap variable required for Linux), while the 16kHz file is in little endian format (no swap variable required for Linux).

The installation of the code can be checked by running the following commands:

NB: replace all references to 8 or 08 with 16 for 16kHz data, and replace all references to Linux with SunOS for a SunOS installation.

AdvancedFrontEnd/linux/bin/AdvFrontEnd *Test_Data/v10770c7.it1.08* *Test_Data/test.mfc* -F RAW -fs 8 -swap
diff *Test_Data/test.mfc* *Test_Data/Reference_Files/processed_08_Linux.mfc*
diff *Test_Data/test.vad* *Test_Data/Reference_Files/processed_08_Linux.vad*

Coder_Decoder/linux/bin/coder_VAD *Test_Data/test.mfc bitstream* *Test_Data/quantised.mfc* –freq 8 –VAD
diff *Test_Data/quantised.mfc* *Test_Data/Reference_Files/quantised_08_Linux.mfc*

DerivCalc/linux/bin/derivCalc *Test_Data/quantised.mfc* *Test_Data/quantised_with_derivatives.mfc Test_Data/test.vad* -COMB -FD
diff *Test_Data/quantised_with_derivatives.mfc Test_Data/Reference_Files/quantised_with_derivatives_08_Linux.mfc*