Manual of configuration and socket communication protocols for online processing using Svtools and array-related programs

By Carlos Ishi, last revision 2020.11.02

General remarks:

The program “carlos-svtools.exe” includes several functions for recording, processing, playing, sending/receiving, labeling and visualizing single and multi-channel audio signals. In this manual, configuration and protocols for part of the functions are described, focusing on online processing.

All configuration files have to be in a “config” directory placed in the same directory of the .exe file.

For all items in the configuration file, the line right below a specified item (e.g., [audio port]) is the valid one. All other lines are ignored.

1. Audio tracker

Program:
carlos-svtools.exe (audio tracker mode) (last update: 2019.06.06)
To start the audio tracker mode:
menu bar: Array – Online audio tracker
tool bar: record button

Related configuration files:
config¥config-controllersocket.txt (socket/save configuration for receiving sound localization and human position)
config¥config-multiarrayspeechactivity.txt (speech activity detection configuration)
config¥config-soundsepsocket.txt (socket/save/play configuration for receiving separated sounds)
config¥roomsensors.txt (array sensor position and orientation configuration)
config¥roomgeometry.txt (room viewer configuration)
config¥config-svtools.txt (GUI configuration)

Description:
This program receives sound direction information from multiple microphone arrays (soundloc) and human positions from the human tracker (layer 2), and outputs sound activities for each human, through TCP socket, according to the following configuration and protocol.
This program also receives multiple audio signals from single microphones (recsound) or separated signals from microphone arrays (soundloc), and re-send them to ASR (julius) or prosody modules (recsound) or remote players, through TCP sockets.

Important: **DO NOT FORGET TO SYNCHRONIZE THE CLOCKS OF ALL PCs**, INCLUDING HUMAN TRACKER RELATED PCs, OTHERWISE AUDIO TRACKER MAY NOT WORK PROPERLY!!!

1.1. Output of speech activity information

This section describes the configurations and protocols for output speech activity information.

Configuration file:
config¥config-controllersocket.txt

```
# port number
[audio track port]
7890
```

Protocol (last update: 2015.07.06):
One line in text format for each human.
The packet size is 1460, so it is completed with '¥0' after the '¥n' delimiter. (From 2016.07.06, the packet size was changed to send only the data size, without zero padding.)
'-10000' indicates non-available data.

```
%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d¥n
```

- timestamp [s] <long> (Unix time)
- timestamp [ms] <short> (Unix time in ms)
- index <int> (countdown index for the humans present at this timestamp)
- human Id <long> (same human Id received from human tracker/layer 2)
- estimated voice activity <int> (-1: not active, 0 ~: active; estimation latency around 100 ~ 200ms)
- estimated sound source height [mm] <int> (mouth height for humans)
- estimated head direction [deg*10] <int> (azimuth angle; available when two or more arrays are used)
- estimation delay [ms] <int> (delay between human tracker data and speech activity data)
- estimated power [dB*10] <int> (directivity power: MUSIC power)
- estimated pitch [Hz*10] <int> (if not available, it is set to -10000)
- estimated pitch reliability <int> (if not available, it is set to -10000)
- estimated lip height command (0 ~ 255) <int> (if not available, it is set to -10000)
- estimated lip width command (-127~127) <int> (if not available, it is set to -10000)
- estimated head pitch command (0~255) <int> (if not available, it is set to -10000)
- unique Id <long> (human unique Id, which is associated with the separated speech channel) (included from 2019.04.08)
- local human Id <long> (local human Id, specified by the audio tracker) (included from 2019.06.06)

Remarks:
Some of the slots are filled with -10000 (non-available number) to make compatible with the output format of other prosody/motion modules, when communicating with Layer2. This protocol might be revised soon.

1.2. Output of separated signals

This section describes the configurations and protocol for output separated signals to ASR (Julius), prosody modules or remote players (recsound, prosody modules). From 2016.04.20, up to 4 audio signals for the first four selected humans can be sent. From 2016.07.06, the program was modified to allow multiple connections for the same separated audio signal.

Configuration file:
config¥config-controllersocket.txt

set the following five items, for each audio receiver (from 2016.07.06):
Port,Host,Channel,Format,ActiveIntervalsOnly
Port: port number of the audio receiver.
Host: empty or “no”, if audio receiver is a client.
Channel: separated signal channel to be sent (currently, it is limited to between 1 and 4)
Format: must be “JULIUS” to send audio to adintool.exe, and “WAV16” to send audio to a prosody module.
ActiveIntervalsOnly: 0 -> send all audio packets; 1 -> send only detected speech intervals. (it must be 1 for JULIUS format)
[audio out info]
5004,erato-mic1,4,JULIUS,1
[audio out info]
4004,erato-mic1,4,WAV16,1

Protocol:
The server will send the specified audio channel with the specified format to each audio receiver, according to the following packet format.
Packets of header + audio data, in binary little endian format.

Packet format for “wav” (last update: 2015.07.06):

4 bytes for packet index <long> (packet index received from capture)
320 bytes of audio data <short> (2 bytes * 160 samples)

Packet format for “julius” (last update: 2015.07.06):

4 bytes for data size <long> [bytes]
N bytes of audio data <short> (2 bytes * N/2 samples)

Packet format for “wav16” (last update: 2016.07.06):
16 bytes for header, 320 bytes for audio data:

4 bytes for time stamp [s] <long> (time stamp received from capture)
2 bytes for time stamp [ms] <short> (time stamp received from capture)
4 bytes for packet index <long> (packet index received from capture)

4 bytes for human id <long> (received from human tracker/layer2)
2 bytes for speech activity [0/1] <short> (speech activity)
320 bytes of audio data <short> (2 bytes * 160 samples)

Remarks:

1.3. Optional automatic start mode

This section describes the configurations to automatically start the audio tracker, without the need of pressing buttons.

Configuration file:
config¥config-svtools.txt

```
# --- Automatic start mode setting
# 0 -> manual start mode
# 3 -> online audio tracker mode
[auto start mode]
3
```

Remarks:
This is equivalent to select the menu “Array – Online audio tracker”, and press the record button to start processing.
It waits for a few seconds to connect to the other modules, before sending the START command and showing the GUI display.
When using the automatic start mode, it is recommended that all other modules are already launched, since the purpose of the START commands sent by the audio tracker is to synchronize all modules.

1.4. Input of separated signals

This section describes the configurations to receive separated speech signals from a sound localization/separation module (soundloc.exe).

Configuration file:
config¥config-soundsepsocket.txt

```
# --- audio socket for receiving audio data from sound localization/separation module
# host name or ip address
```

```
[audio host]
localhost
# port number
[audio port]
7994
# audio format
# bin (separated signals), no (no transmission)
[audio format]
bin
# number of channels for audio transmission/reception (needs revision)
[audio channels]
4

# --- desired sound separation mode (initial configuration; can be changed by the audio tracker GUI):
# ds: delay-sum beamformer
# wfds: wiener pre-filtering followed by delay-sum beamformer
# dsbm: delay-sum beamformer followed by interchannel suppression
# wfdsbm: wiener pre-filtering followed by delay-sum beamformer, and followed by interchannel suppression
[soundsep mode]
dsbm
```

Remarks:

If the system is running in an environment where the predominant background noise is stationary (such as air conditioners and PC fan noises), it is recommended to use wfds or wfdsbm modes. Otherwise, if the background noise is non-stationary (such as babble noise in open house environments), it is recommended not to use the "wf" mode.

The latest version of Julius (2016.03) has training models adapted to the "ds" and "wfds" modes. Although the "bm" modes provide better separation of overlapped sounds (for example, it is often to have overlaps between ERICA and visitor voices), Julius is still not adapted to the signals with strong distortions after the inter-channel suppression processing.

Currently (2016.04.21) the audio tracker can receive separated signals from only one microphone array. Extensions for other arrays are coming soon.

Related references:

- 石井カルロス寿憲, エヴァン・イアニ, 萩田紀博 (2016). 複数のマイクロホンアレイによる音源方向情報と人位置情報に基づく音声区間検出および顔の向きの推定の評価, 日本ロボット学会誌, Vol.34 No.3, 39-44, April 2016.
- Ishi, C., Even, J., Hagita, N. (2015). "Speech activity detection and face orientation estimation using multiple microphone arrays and human position information," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015), pp. 5574-5579, Sep., 2015.

- Ishi, C., Even, J., Hagita, N. (2014) Integration of multiple microphone arrays and use of sound reflections for 3D localization of sound sources. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E97-A, No.9, pp.1867-1874, Sep. 2014.

2. Recsound or Prosody module: sound recorder, voice activity detection, prosodic information extraction and motion generation (for single-channel audio signals)

Program:
carlos-svtools.exe (recsound or prosody mode) (last update: 2020.11.02)
or
carlos-lipsync.exe (old version)
To start the recsound mode:
tool bar: record button (for the default analysis view mode)
or
START button (for the teleoperation prosody motion view mode)

Related configuration files:
config¥config-recsound.txt (socket/save/player configuration)
config¥config-sigproc.txt (signal processing configuration)
config¥config-lipsync.txt (lip/head/face motion generation configuration)
config¥config-svtools.txt (GUI configuration)

Description:
This program captures audio signal for a single source from a local device (microphone) or receives audio signal for a single source from a socket, optionally saves the input audio signal to files, optionally sends and/or plays the input audio signal, and optionally outputs voice activity, and/or prosodic information, and/or motion generation information, through TCP socket, according to the following configuration and protocol.

2.1. Output of voice activity, prosodic information and motion generation commands

This section describes the configurations and protocol for sending voice activity, prosodic information and motion generation commands.

Configuration file:
config¥config-recsound.txt

```
# prosody/motion output format:
# robovie -> socket for sending head motion control parameters to a remote robot (robovie)
# android -> socket for sending lip/head motion control parameters to a remote android 101 (HI1), 102 (F)
# telenoid -> socket for sending lip/head motion control parameters to a remote telenoid 103
# iwai -> socket for sending lip motion control parameters to a remote telenoid server using iwai's protocol
```

```
# layer2 -> send prosodic and motion control parameters to a remote layer2 client
[motion format]
layer2
# host name or ip address: "no", for server
[motion host]
no
[motion port]
12000
# delay (in ms) for output prosody or motion data, relative to the captured/received time
# minimum delay is 60 ms, necessary for processing/smoothing.
[motion delay]
60
```

Protocol for "layer2" mode (last change: 2016.05.19):
One line in text format for each human id.
The packet size is 1460, so it is completed with '¥0' after the '¥n' delimiter.
'-10000' indicates non-available data.

```
%d,%d,0,0,%d, -10000,-10000,0,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d¥n
```

- timestamp [s] <long> (Unix time equivalent to the captured/received instant)
- timestamp [ms] <short> (Unix time equivalent to the captured/received instant)
- index <int> (set to 0, since only one source is managed)
- human Id <long> (currently set to 0, if a human id can be attributed for each microphone, this id can be sent in future)
- estimated voice activity (0/1) <int> (voice activity estimated from signal power; latency around 80 ~ 100ms?)
- estimated mouth height [mm] <int> (not available, currently set to -10000)
- estimated head direction [deg*10] <int> (not available, currently set to -10000)
- estimation delay [ms] <int> (not used, currently set to 0)
- estimated power [dB*10] <int> (input signal power)
- estimated pitch [Hz*10] <int> (input signal F0)
- estimated pitch reliability <int> (0 ~ 14, larger than 11 is reliable)
- estimated lip height command (0 ~ 255) <int> (formant-based estimation; 0: maximally closed, 255: maximally open)
- estimated lip width command (-127~127) <int> (formant-based estimation; -127: maximally rounded, 127: maximally stretched)

- estimated head pitch command (0~255) <int> (F0-based estimation; 0: maximally facing down, 255: maximally facing up) (up to this item, defined in 2015.07.06)
- estimated vowel probability [0/1] <short> (formant-based vowel probability) (from this item, included from 2016.05.19)
- estimated first formant [Hz] <short> (input signal F1)
- estimated second formant [Hz] <short> (input signal F2)
- estimated third formant [Hz] <short> (input signal F3, less reliable)
- estimated forth formant [Hz] <short> (input signal F4, less reliable)
- configured center vowel F1 [Hz] <short> (center vowel F1 configured by the user, adjusted by GUI)
- estimated periodicity (0 ~ 1000) <short> (periodicity based on auto-correlation peak) (included from here: 2020.11.02)
- estimated sonorant power [dB*10] <short>
- estimated high power [dB*10] <short> (power of high-freq. band)
- estimated mid power [dB*10] <short> (power of mid-freq. band)
- estimated low power [dB*10] <short> (power of low-freq. band)
- estimated f0 [semitone * 10] <short> (F0 after post-processing, in semitones * 10)
- estimated delta-f0 [semitone * 10] <short> (first regression coefficient for F0 curve, in semitones * 10)
- estimated A1-A3 [dB*10] <short> (difference of power between F1 and F3 bands)
- estimated F1F3syn [-1000 ~ 1000] <short> (related to voicing features of F1 and F3 bands, for breathiness characterization)
- estimated H1 – A1 [dB*10] <short> (difference of power around the first harmonic and the first formant: related to low freq. loss in pressed voices)
- estimated breathiness power [dB*10] <short> (breathiness power relative to the maximum power of the utterance)
- estimated H1 – A3 [dB*10] <short> (difference of power around the first harmonic and the third formant; related to spectral tilt; related to vocal tension)
- estimated IPS [-1000 ~ 1000] <short> (inter-pulse similarity measure for discriminating creaky voice from impulsive noises)
- estimated NACR [-1000 ~ 1000] <short> (auto-correlation peak ratio; related to double-periodicity in creaky and harsh voices)
- estimated TLD [] <short> (auto-correlation time-lag difference; related to double-periodicity in creaky and harsh voices)
- estimated WR [-1000 ~ 1000] <short> (auto-correlation peak width ratio; related to double-periodicity in creaky and harsh voices)
- estimated APE [-1000 ~ 1000] <short> (adaptive pre-emphasis coefficient; related to spectral tilt; related to vocal tension)

Remarks:
Some of the slots are filled with -10000 (non-available number) to make compatible with the output format of other audio tracker module, when communicating with Layer2.
Be careful that in the case the input signal is received by socket, the time stamp might not be exactly equivalent to the captured time, due to variations in the network transmission time.
[motion format], [motion host], [motion port] are equivalent to [robovie format], [robovie host], [robovie port] in the previous versions.
[motion delay] can be set to 0, in offline processing, when recorded audio files are already processed.

2.2. Optional control from remote controller socket

This section describes the configurations for optionally receiving START/STOP commands from a remote controller socket, in order to synchronize the recording time in multiple modules.

Configuration file:
config¥config-recsound.txt

```
# ----- controller socket configuration
# controller format
# no: no transmission with controller server
# sync: receives START/STOP commands from a controller server, in order to synchronize multiple captures
[controller format]
sync
# host name or IP (when using controller)
[controller host]
localhost
# port number (when using controller)
[controller port]
20101
```

Protocol for the controller socket (last change 2013.02):
After connection with the controller server, the program waits for receiving commands in text format.

```
START¥tTimestamp¥tMsec¥n
STOP¥tTimestamp¥tMsec¥n
```

- Timestamp is the Unix time in seconds.
- Msec is the Unix time in msec.

If a START command is received, the program starts saving audio packets. The received start time stamps are used to create the file names, if the audio signals are saved. (See Section 2.4)
If a STOP command is received, the program stops saving audio packets.

2.3. Optional input/output of audio signals

This section describes the configurations to optionally send and receive audio signals through socket transmission. For example, in one side a recsound module can be configured to capture and send audio signals, while in the other side another recsound module can be configured to receive the audio signals. In this case, audio signals can be processed to output prosodic or motion data, in either capture side or receiver side. Another usage is to configure the recsound module to receive a separated signal (after microphone array processing) from the audio tracker module, and process to output prosodic or motion data.

Configuration file:
config¥config-recsound.txt

Receiver side:

```
# input format:
# audio input configuration
# mic -> capture audio data from local device (mic)
# wav -> socket for receiving audio data from a remote capture server (12 bytes header)
# fsync -> socket for receiving audio data from a remote capture client (ATRASR format: 4 bytes header)
# julius -> socket for receiving audio data from a remote capture server (Julius format: 4 bytes header)
# wav16 -> socket for receiving time stamped audio data from a remote capture server (Audio tracker format: 16 bytes header) (2016.07.06)
[input format]
wav16
# host address/name and port number (when [input format] is not set to "mic")
[input host]
localhost
[input port]
4001
# number of input channels: 1 for mono, 2 for stereo (has to be the same of the audio server)
[input channels]
1
# sampling rate (in Hz)
# default for parameter extraction: 16000 Hz
```

other sampling rates can be set for recording purposes; however they are not adapted for parameter extraction
[input sampfreq]
16000

Capture side:
audio output configuration
no -> do not output audio data
fsync -> socket for sending audio data to remote ATRASR
wav -> socket for sending audio data to a remote player
julius -> socket for sending audio data to a remote Julius
wav16 -> socket for receiving time stamped audio data from a remote capture server (Audio tracker format: 16 bytes header) (2016.07.06)
[output format]
wav
host name or ip address ("no", for server)
[output host]
no
[output port]
4001

Protocols:
The audio server will send the required audio channel with the required format to the audio client, according to the following packet format.
Packets of header + audio data, in binary little endian format.

Packet format for "wav" (last update: 2015.07.06):

4 bytes for packet index <long>
320 bytes of audio data <short> (2 bytes * 160 samples)

Packet format for "julius" (last update: 2015.07.06):

4 bytes for data size <long> [bytes]
N bytes of audio data <short> (2 bytes * N/2 samples)

Packet format for "wav16" (last update: 2016.07.06):
16 bytes for header, 320 bytes for audio data:

4 bytes for time stamp [s] <long> (time stamp received from capture)
2 bytes for time stamp [ms] <short> (time stamp received from capture)
4 bytes for packet index <long> (packet index received from capture)
4 bytes for human id <long> (received from human tracker/layer2)
2 bytes for speech activity [0/1] <short> (speech activity)
320 bytes of audio data <short> (2 bytes * 160 samples)

Remarks:
These formats are compatible with the output audio signals from other recsound modules running in remote machines for capturing individual sound sources (by a single close-talk microphone), as well as from the audio tracker module providing the separated signals (after microphone array processing) in individual ports.
Be careful to set one side as server and the other side as client, and set the same audio packet format in both sides.
Currently only one client/server pair can be connected (2016.04.20). In future, the program might be modified to allow connection by multiple clients.

2.4. Options for saving audio signal to files

This section describes the configurations of formats to save the input audio files.

Configuration file:
config¥config-svtools.txt

```
# save configuration
# no: do not save
# wav: WAVE format (little endian)
# raw: headerless format (big endian)
[save audio format]
wav
# directory for saving audio signals
[save audio dir]
c:¥carlos¥data¥recsound¥
# time interval for saving (in ms)
# 0 -> do not cut files
# 60000 -> cut files every minute (60000 ms = 1 minute)
# 300000 -> cut files every 5 minutes (300000 ms = 5 minutes)
# -1 -> cut files according to controller commands
[save time interval ms]
```

300000

Remarks:
The save directory has to be created manually. If the directory does not exist, files will not be saved.
A subdirectory with the format YYYYMMDD is created in the save directory.
File names with the format YYYYMMDD-HHMMSS-MMM.wav are saved.
If the controller socket is connected ([controller format] is set to “sync”), the first file name is obtained from the time stamp received with the START command (see Section 2.2).
Otherwise ([controller format] is set to “no”), the time stamp of the instant the module is launched is used for the file name.
If the time interval is larger than 0, files will be cut in time stamps with integer multiples of the time interval. For example, when the file is cut each 60000ms (1 minute), file names will have the format YYYYMMDD-HHMM00-000.wav, while for 300000ms (5 minutes), file names will have the format YYYYMMDD-HHM0-000.wav or YYYYMMDD-HHM5-000.wav
If the time interval is set to -1, files will be cut every time a START command is received from the controller socket.

2.5. Options for save/compute parameter

This section describes the configurations of options for computing and saving parameters (frame feature extraction, each 10ms).

Configuration file:
config¥config-svtools.txt

```
# save/compute parameters (feature extraction)
# 0 -> don't save, don't compute (default)
# 1 -> save, compute
# -1 -> don't save, compute
[save param]
-1
[save param dir]
c:¥svtools¥param¥
```

Remarks:
For only recording audio signals (without the need of processing prosody or motion data), set [save param] to 0.
For output prosody or motion data (without the need of saving the extracted parameters), set [save param] to -1.

If you want to save the parameters to file, create the param directory specified in [save param dir]. If the directory does not exist, files will not be saved.

2.6. Optional player for input audio signal

This section describes the configurations for optionally playing the input audio signal.

```
# input player flag
# 0: do not play
# 1: play
[input player]
1
# input player delay (in ms), necessary for buffering
# 10 for local audio server (mic)
# 50 for remote wired audio server
# 100 for remote wireless audio server
[input player delay]
80
# input player gain (linear gain)
# 0 ~ 1: attenuation
# 1 ~: amplification
[input player gain]
1
# input player balance
# -100: left only
# 100: right only
# 0: mono
[input player balance]
0
# player device id
# -1: use windows default device
[input player device]
-1
```

2.7. Optional automatic start mode and viewer settings

This section describes the configurations to automatically start the recsound module, without the need of pressing record/start buttons.

Configuration file:

config¥config-svtools.txt

```
# --- Automatic start mode setting
# 0 -> manual start mode
# 1 -> online recsound mode
[auto start mode]
1
# --- Initial view mode setting
# 0 -> normal analysis view mode
# 1 -> teleoperation (prosody motion) view mode
[teleop mode]
1
```

Remarks:
[auto start mode] set to 1 is equivalent to press the record button (in the tool bar of the analysis view) or the start button (in the teleoperation view) to start the online processing.
[teleop mode] set to 1 is equivalent to press the "TELE VIEW" button in the tool bar of the analysis view.
If you have enough CPU operating ratio remaining, switch to the analysis view to check the temporal features of the extracted parameters. The detected speech intervals can also be easily viewed, making the power threshold easier to be adjusted.

2.8. Signal processing parameter settings

This section describes the configurations of parameters related to signal processing during prosodic feature extraction.

Configuration file:
config¥config-sigproc.txt

```
# power threshold (in dB) for speech activity detection
# should be adjusted according to mic gain and level
# -10000: automatic power threshold (using first intervals: 0.3 to 1 seconds),
#           but preferable to adjust fixed values for fixed environments
# default = 0 dB
[power threshold]
-10000

# F0 estimation method
```

```
# 0 -> ACF of LPC residual signal (recommended for close talk mics with high SNR)
# 1 -> ACF of low frequency region (F1 band)
# 3 -> ACF of mid frequency region (F3 band)
# 6 -> SACF (recommended for noisy and distorted signals, after array processing)
[f0 estimation]
6
0

# option for double pre-emphasis in the input signal
# 0: single pre-emphasis (default)
# 1: double pre-emphasis (recommended for high-pitch female and children voices, and whispery/breathy voices)
[double pre-emphasis]
0

# -1 -> default = 19 coefficients
# increasing the lpc dimension may improve formant peak extraction in some cases
[lpc dimension]
-1
```

Remarks:

The double pre-emphasis option for female and children voices might improve formant extraction used for lip motion generation. However, it might degrade F0 estimation used for intonation and paralinguistic information extraction.

Related references (prosody):

- Ishi, C., Dong, L., Ishiguro, H., and Hagita, N. (2011). "The effects of microphone array processing on pitch extraction in real noisy environments," Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), 550-555.
- Ishi, C.T., Ishiguro, H., Hagita, N. (2010). Analysis of the roles and the dynamics of breathy and whispery voice qualities in dialogue speech. EURASIP Journal on Audio, Speech, and Music Processing 2010, ID 528193, 1-12 Jan. 2010.
- Ishi, C.T., Ishiguro, H., Hagita, N. (2008). Automatic extraction of paralinguistic information using prosodic features related to F0, duration and voice quality. Speech Communication 50(6), 531-543, June 2008.
- Ishi, C.T., Sakakibara, K-I., Ishiguro, H., Hagita, N. (2008). A method for automatic detection of vocal fry. IEEE Transactions on Audio, Speech and Language Processing, Vol. 16, No. 1, 47-56, Jan. 2008.
- Ishi, C.T. (2006), The functions of phrase final tones in Japanese: Focus on turn-taking. Journal of Phonetic Society of Japan, Vol. 10 No.3, 18-28, Dec. 2006.
- 石井カルロス寿憲, 榊原健一, 石黒浩, 萩田紀博 (2006) Vocal Fry 発声の自動検出法. 電子情報通信学会論文誌DVol. J89-D, No. 12, 2679-2687, Dec. 2006.
- 石井カルロス寿憲，石黒浩，萩田紀博 (2006) 韻律および声質を表現した音響特徴と対話音声におけるパラ言語情報の知覚との関連. 情報処理学会論文誌 Vol. 47, No. 6, 1782-1793, June 2006.
- Ishi, C.T. (2005) Perceptually-related F0 parameters for automatic classification of phrase final tones. IEICE Trans. Inf. & Syst., Vol. E88-D, No. 3, 481-488
- Ishi, C.T., Hirose, K. & Minematsu, N. (2003). Mora F0 representation for accent type identification in continuous

speech and considerations on its relation with perceived pitch values. Speech Communication, Vol. 41, Nos. 2-3, 441-453

Related references (motion generation):

- Ishi, C., Ishiguro, H., Hagita, N. (2013). Analysis of relationship between head motion events and speech in dialogue conversations. Speech Communication 57 (2014) 233–243, June 2013.
- 石井カルロス寿憲, 劉超然, 石黒浩, 萩田紀博 (2013). 遠隔存在感ロボットのためのフォルマントによる口唇動作生成手法, 日本ロボット学会誌, Vol. 31, No. 4, 83-90, May 2013.
- Ishi, C., Liu, C., Ishiguro, H. and Hagita, N. (2012). "Evaluation of a formant-based speech-driven lip motion generation," In 13th Annual Conference of the International Speech Communication Association (Interspeech 2012), Portland, Oregon, pp. P1a.04, September, 2012.
- 劉超然, 石井カルロス寿憲, 石黒浩, 萩田紀博 (2013). 人型コミュニケーションロボットのための首傾げ生成手法の提案および評価, 人工知能学会論文誌, vol. 28, no. 2, pp. 112-121, January, 2013.
- Liu, C., Ishi, C., Ishiguro, H., Hagita, N. (2013). Generation of nodding, head tilting and gazing for human-robot speech interaction. International Journal of Humanoid Robotics (IJHR), vol. 10, no. 1, January, 2013.

3. Sound localization and separation (for individual microphone arrays)

Program:
soundloc.exe (last update: 2016.06.25)

Related configuration files:
config¥config-arraysocket.txt (sensor id, socket/save configuration)
config¥config-soundlocalization.txt (sound localization processing parameter configuration)
config¥config-soundseparation.txt (sound separation processing parameter configuration)
config¥sensor-XXXX.txt (geometry of the microphone array)

Description:
This program receives multi-channel array signals, and outputs sound localization (sound direction) results and/or sound separation results (audio signals), through TCP socket, according to the following configuration and protocol. Details on the protocols are only necessary for program developers.

3.1. Output of sound localization results

This section describes the configurations for sending sound localization results to multiple clients.

Configuration file:
config¥config-arraysocket.txt

```
# --- sound localization output socket
[soundloc port]
7992
[soundloc format]
data
[sensor id]
11001
```

Protocol:
One line in text format for each detected sound source.
The packet size is 1460, so it is completed with '¥0' after the '¥n' delimiter.

```
SLOCM¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥n
```

- sensor Id <long> (set in the configuration file)

- timestamp [s] <long> (Unix time received by array capture)
- timestamp [ms] <int> (Unix time received by array capture)
- packet index <long> (packet index received by array capture)
- detected source index <int> (countdown index for active sources detected at this time: “0” for the last source. If no sound sources are detected, “-1” is sent.)
- estimated source azimuth angle [deg*10] <int>
- estimated source elevation angle [deg*10] <int>
- estimated source range [mm] <int> (the source range is computed, but might not be reliable, since MUSIC power is not as sharp as in direction estimation.)
- detected source MUSIC power [dB*100] <int>
- delay [ms] <int> (latency between the captured time and the instant the localization results are ready to be sent)

3.2. Output of sound separation results (from individual microphone arrays)

This section describes the configurations for sending multiple (currently up to 4, after 2016.03.06) separated audio signals to the audio tracker module.
Currently (after 2016.04.20), separated signals are provided by the audio tracker module, so users do not need to care about details on the communication protocol.

Configuration file:
config¥config-arraysocket.txt

```
# --- soundsep socket
# for sending separation data to multiple clients
[soundsep port]
7994
# format to send sound separation results
# no: do not send
# data: binary data for audio packet for each separated signal
[soundsep send format]
data
# sound separation mode
# ds: only Delay-Sum beamforming (DS)
# wfds: background stationary noise suppression + DS
# wfdsbm: above + interchannel suppression
# dsbm: DS + interchannel suppression
[soundsep send mode]
dsbm
```

Protocol (last update: 2015.12.24):
Binary format: 358 bytes per packet of 10ms audio data, in little endian.
38 bytes for header + 320 bytes for audio data
Packet format:
2 bytes for sensor id <short>
4 bytes for time stamp (s) <long>
2 bytes for time stamp (ms) <short>
4 bytes for packet index <long>
2 bytes for azimuth <short> [degrees*10]
2 bytes for elevation <short> [degrees*10]
2 bytes for range <short> [mm]
2 bytes for source id <short>
2 bytes for suppression flag <short> // 2015.10.23 included to notify if selected human is target or anti-target
4 bytes for human id <long> // 2015.12.24 included to send human data: id and position
4 bytes for human pos_x <long> [mm]
4 bytes for human pos_y <long> [mm]
4 bytes for human pos_z <long> [mm]
320 bytes for audio data <short> (2 bytes * 160 samples)

Remarks:
The current version (from 2016.03.06) can provide up to 4 separated signals, corresponding to the humans selected in the audio tracker module.

3.3. Communication with remote controller socket (audio tracker)

This section describes the configurations for receiving commands from a remote controller socket, and sending localization results.
If a controller socket is configured, the program waits for START/STOP commands to save sound localization and separation results to files, sends localization results to the controller, and receives commands to separate signals for target and anti-target sound sources.
Details on the protocols are omitted since they are only necessary for developers of soundloc and audio tracker modules. Users do not need care about details on this protocol.

Configuration file:
config¥config-arraysocket.txt

```
# ----- controller socket configuration
# controller format
```

no: no transmission with controller server
data: receives START/STOP commands from a controller server, and send sound localization results
[controller format]
data
host name or IP (when using controller)
[controller host]
localhost
port number (when using controller)
[controller port]
20102

Protocol for receiving commands from the controller socket (last update 2013.02):
After connection with the controller server, the program waits for receiving commands in text format.

START¥tTimestamp¥tMsec¥n
STOP¥tTimestamp¥tMsec¥n

- Timestamp is the Unix time in seconds.
- Msec is the Unix time in msec.

If a START command is received, the program starts capturing, sending and/or saving audio packets.
The received start time stamps are used to create the file names, if the audio signals are saved.
If a STOP command is received, the program stops capturing, sending and/or saving audio packets.

Commands to separate signals for target and anti-target sound sources (when humans are selected in the audio tracker GUI).

SLOC1H1¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%.2f¥t%.2f¥t%.2f¥t%d¥t%d¥n

Commands to separate strongest directional signal (when no human is selected in the audio tracker GUI).

SLOCM¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥t%d¥n

Protocol for sending sound localization results to the controller socket:
The same described in Section 3.1.

Related references:

- 石井カルロス寿憲, 劉超然, Jani Even (2015) “音環境知能技術を活用した聴覚支援システムのプロトタイプの開発”, 第 43 回人工知能学会 AI チャレンジ研究会, Nov. 2015.
- Ishi, C.T., Chatot, O., Ishiguro, H., and Hagita, N. (2009). “Evaluation of a MUSIC-based real-time sound localization of multiple sound sources in real noisy environments,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), 2027-2032.

4. Array capture
(for individual microphone arrays)

Program:
tdusb_capture_server.exe (last update 2016.02.05)

Configuration file:
config¥config-arraysocket.txt

Description:
This program captures multi-channel signals from the TD-BD-16ADUSB device, and sends the multi-channel array signals through TCP socket to multiple clients, according to the following configuration and protocol. The program optionally receives START/STOP commands from a remote controller.

4.1. Output of multi-channel audio signals

This section describes the configurations and protocol for sending multi-channel audio signals for multiple clients.

Configuration file:
config¥config-arraysocket.txt

```
# ----- audio socket configuration
# audio format
# no: no transmission (when the program is used only for recording multi-channel data)
# bin: binary format
[audio format]
bin
# audio port number
[audio port]
7998
```

Protocol for the audio socket (last update 2013.02):
After connection with an audio client, the program waits for receiving the channel numbers to be sent. For example, if the client wants to receive the whole 16 channels, the client has to send:

```
channels=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16¥n
```

For sub-array processing, the client can receive some of the channels, for example by sending:

channels=1,5,9,13¥n

If the [controller format] is “sync” (see Section 4.2), the program waits for connecting to a controller server and receiving a “START” command to start sending the audio packets to the audio clients.
If the [controller format] is “no”, the program starts sending the audio packets of the selected channels for each audio client, right after receiving the channel numbers.

Audio packets are sent in binary format for each 10ms audio data.
12 bytes for header, followed by N*320 bytes for audio data, where N is the number of selected channels to be sent. (little endian)
Packet format:
2 bytes for sensor ID <short>
4 bytes for time stamp (s) <long>
2 bytes for time stamp (ms) <short>
4 bytes for packet index <long>
320 bytes of audio data for the first channel <short> (2 bytes * 160 samples)
320 bytes of audio data for the second channel <short> (2 bytes * 160 samples)
…
320 bytes of audio data for the last channel <short> (2 bytes * 160 samples)

Remarks:
Sensor ID is sent, but not being used, since sensor id is currently being attributed for soundloc modules.
Packet index is an increasing integer number for the number of captured 10ms frames. It can be used to check missing data at the network transmission level. However, currently it does not account for missing data at the capture level (USB transmission).

4.2. Optional control from remote controller socket

This section describes the configurations for optionally receiving START/STOP commands from a remote controller socket, in order to synchronize multiple modules.

Configuration file:
config¥config-arraysocket.txt

----- controller socket configuration
controller format
no: no transmission with controller server

sync: receives START/STOP commands from a controller server, in order to synchronize multiple captures
[controller format]
sync
host name or IP (when using controller)
[controller host]
localhost
port number (when using controller)
[controller port]
20101

Protocol for the controller socket (last update 2013.02):
After connection with the controller server, the program waits for receiving commands in text format.
START¥tTimestamp¥tMsec¥n
STOP¥tTimestamp¥tMsec¥n

- Timestamp is the Unix time in seconds.
- Msec is the Unix time in msec.

If a START command is received, the program starts capturing, sending and/or saving audio packets. The received start time stamps are used to create the file names, if the audio signals are saved.
If a STOP command is received, the program stops capturing, sending and/or saving audio packets.

5. Applications for voice activity detection (VAD) using the microphone arrays
音声アクティビティ検出（VAD）への応用法

There are a couple of methods for obtaining VAD information. The following methods are in order of preference.
音声アクティビティ検出（VAD）への応用として、いくつかの方法があります。順番はお勧めする順となっています。

5.1. When multiple microphone arrays (2 or more) and human tracker are available
音源定位（アレイ 2 個以上)、人位置情報有りの場合

Speech activity information is obtained for each human id in the environment.
Audio tracker から、各人の activity を取得するのがいいです。

config¥config-multiarrayspeechactivity.txt
[detection type]
2

Problems: If the audio source is not facing to at least two arrays, speech activity might fail.
問題点 : アレイを背にした状態では音源は検出され難い。ぼそぼそ（声を張らない）発話は検出され難い。

5.2. When only one microphone array and human tracker are available
音源定位（アレイ 1 個)、人位置情報有りの場合

Speech activity information is obtained for each human id in the environment.
Audio tracker から、各人の activity を取得します。

config¥config-multiarrayspeechactivity.txt
[detection type]
1

Problems: If the audio source is not facing to the array, speech activity will not be detected. If two audio sources (two humans) are in the same direction relative to the array, both might be detected as active.
問題点 : アレイを背にした状態では音源は検出され難い。アレイに対して、同じ方向に２つの音源が存在する場合、いずれもアクティブと検出されてしまう。ぼそぼそ（声を張らない）発話は検出され難い。

5.3. When only one microphone array and manual human positions are available
音源定位（アレイ 1 個)、疑似人位置情報の場合

When human tracker is not available, human positions (currently at most two positions) can be

manually set in the Audio Tracker GUI, so that speech activity information can be obtained for them.
人位置情報がない場合、疑似的に人位置を手動的に設定して、Audio tracker から、その人の voice activity を取得することができます。

config¥config-multiarrayspeechactivity.txt
[detection type]
1

How to set human positions manually through the GUI:
Mouse left button click while pressing the <SHIFT> key creates a human with id -1.
Mouse right button click while pressing the <SHIFT> key creates a human with id -2.
The human positions can be changed by doing the above in different positions.
To clear the manual human positions, click the mouse left button on one of the arrays.

GUI 上の操作：
シフトを押しながら、マウス左クリックで疑似の人位置が設定できます。
その場合、human id は-1 となる。
シフトを押しながら、マウス右クリックで 2 人目の疑似の人位置が設定できます。
その場合、human id は-2 となる。
人位置を変更したい場合は、シフトを押しながら、マウス左または右クリックをしてください。
疑似人位置を消す場合は、いずれかのアレイをマウス左クリックしてください。

5.4. When only sound direction information is available
音源定位のみ、人位置情報無しの場合

When human positions are not available, and only sound direction information is available from the Soundloc.exe module, receives the sound direction information, and set speech activity to ON if the detected direction is around the target direction (say within a range from 10 degrees), and the MUSIC power exceeds a threshold.
Soundloc.exe から直接音源定位結果を取得して、特定の方向の周辺（例えば誤差 10 度の範囲内）に MUSIC power が閾値を超えていれば、activity を ON にする。

Related references:

- 石井カルロス寿憲, エヴァン・イアニ, 萩田紀博 (2016). 複数のマイクロホンアレイによる音源方向情報と人位置情報に基づく音声区間検出および顔の向きの推定の評価, 日本ロボット学会誌, Vol.34 No.3, 39-44, April 2016.
- Ishi, C., Even, J., Hagita, N. (2015). "Speech activity detection and face orientation estimation using multiple microphone arrays and human position information," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015), pp. 5574-5579, Sep., 2015.
- Ishi, C., Dong, L., Ishiguro, H., and Hagita, N. (2010). "Sound interval detection of multiple sources based on sound directivity," Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), 1982-1987.

6. How to get separated signals from Audio tracker
Audio tracker から分離音を出力する方法

Last revision: 2016.05.19

Program:
carlos-svtools.exe (audio tracker mode) (last update: 2016.05.19)

6.1. Selection of the audio sources to be separated
分離の対象となる音源の選択

For selecting a target source to be separated, click the mouse left button on the target human position. For selecting an anti-target source to be suppressed, click the mouse right button on the anti-target human position.
For removing a selection, click again the mouse left or right button on a selected human position.
Anti-targets have effect only when the audio output mode includes "bm" (inter-channel suppression). The audio output mode can be switched in the Audio tracker GUI menu Play – Soundsep.

ターゲットにしたい（分離音の出力対象となる）人をマウス左クリックで選択します。
アンチターゲットにしたい（妨害音となる）人または音源をマウス右クリックで選択します。
選択を外したい場合は、再度マウス左または右クリックをしてください。
アンチターゲットは、音声出力モードが bm（inter-channel suppression）を含むモードにのみ効果があります。
出力モードは、Audio tracker GUI のメニュー Play – Soundsep から選択できます。

6.2. Analog output (from player device) アナログ出力
The audio signal of a selected target human can be obtained from the analog output of the audio player device. If multiple target humans are selected, the analog output will be a summation of all separated signals, so if the purpose is to forward the signal to a speech recognition module, only one target human should be selected. Multiple selections of anti-target sources are allowed.

Audio Tracker で、選択された人の音声をアナログでオーディオデバイスに出力することができます。ターゲット音が複数選択されている場合は、混ざって出力されるため、音声認識などを目的にした場合は、ターゲット音源を一つのみ選択することになります。アンチターゲット音源は複数選択可能です。

Warning: Depending on the capture/player device and data transmission condition, the separated signal might be intermittent. To avoid that, a sufficiently large delay must be set in the configuration file (hundreds of milliseconds). If the purpose is to send the separated signals to other modules such

as speech recognition or prosodic processing, it is recommended to use the socket transmission described in the next section.

問題点：遅延状況によって、音がブツブツ途切れる場合があります。そのため、十分な遅延を設定する必要があります。目的が音声認識や韻律処理など、他のモジュールに音声データを送信することであれば、次節のソケット通信をお勧めします。

Configuration file:
config¥config-soundsepsocket.txt

```
# --- play input audio
# 0 -> do not play
# 1 -> play
[audio player]
1
# player gain
[audio player gain]
4
# player delay (in ms)
[audio player delay]
250
```

6.3. Socket output ソケット出力

The separated signals of the selected humans can be obtained from socket transmission. Currently (2016.05.19), at most four separated signals (for the first four selected humans) can be output. Both target and anti-target sources are output.

Audio Tracker で、選択された人の音声をソケット通信により出力することができます。現在（2016.05.19)、最大４人の音声（最初に選択された４人）が出力可能となります。
ターゲットにしたい音源（マウス左クリックで選択）と、アンチターゲットにしたい音源（マウス右クリックで選択）のいずれも出力の対象となります。

The configurations of the socket and formats for output the separated signals are described in Section 1.2 (in the Audio tracker chapter).
The output ports are attributed according to the order the audio sources are selected. For example, the separated signal of the first selected human will be sent to the port configured in [audio out1 port], while the one of the third selected human will be sent to the port configured in [audio out3 port].

出力ソケットと出力形式を設定してください。(1.2 節を参照)
音源が選択された順番で、出力ソケットの番号が割り当てられます。例えば、1 番目に選択した人の分離音は[audio out1 port]に設定したポート番号に出力され、3 番目に選択した人の分離音は、[audio out3 port]に設定したポート番号に出力されます。