Current Version Date: Tuesday, November 18, 2008

Contents

1 Hardware

1.1 The Data Acquisition and Stimulation Hardware

FART – Flexible Auditory Rotating Target
RA16
RP2_1
Microcontroller
I2C
Bruel & Kjaer

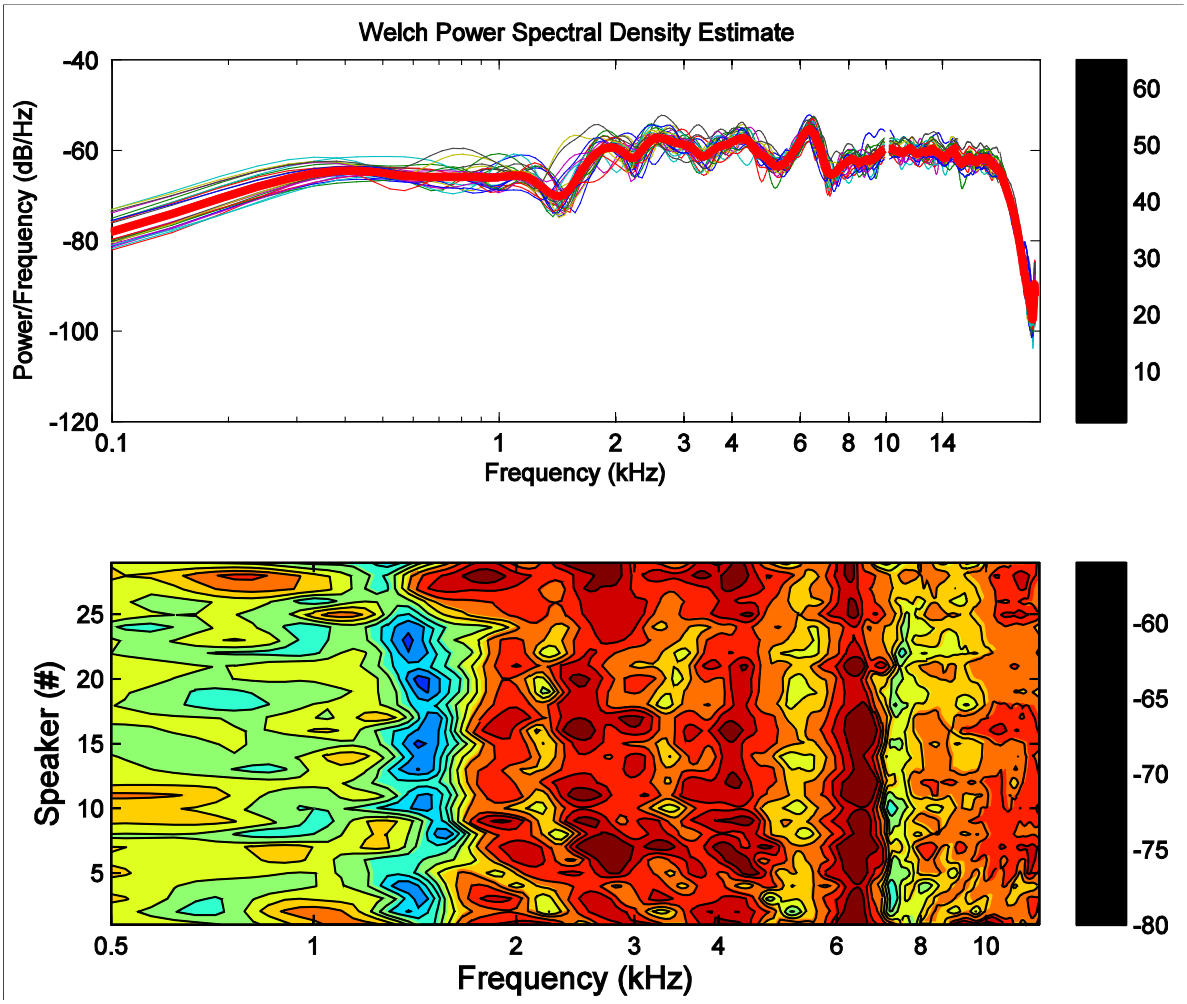

Figure 1. "Speaker Characteristics"of the Hoop in the FART1 experimental room. Measured February 19th, 2008. Measurements were taken near the center of a subject's head. The microphone was pointed towards the speaker by means of a laser. Settings of the Bruel& Kjaer: A-weighted, high-pass at 22.4 Hz, fast.

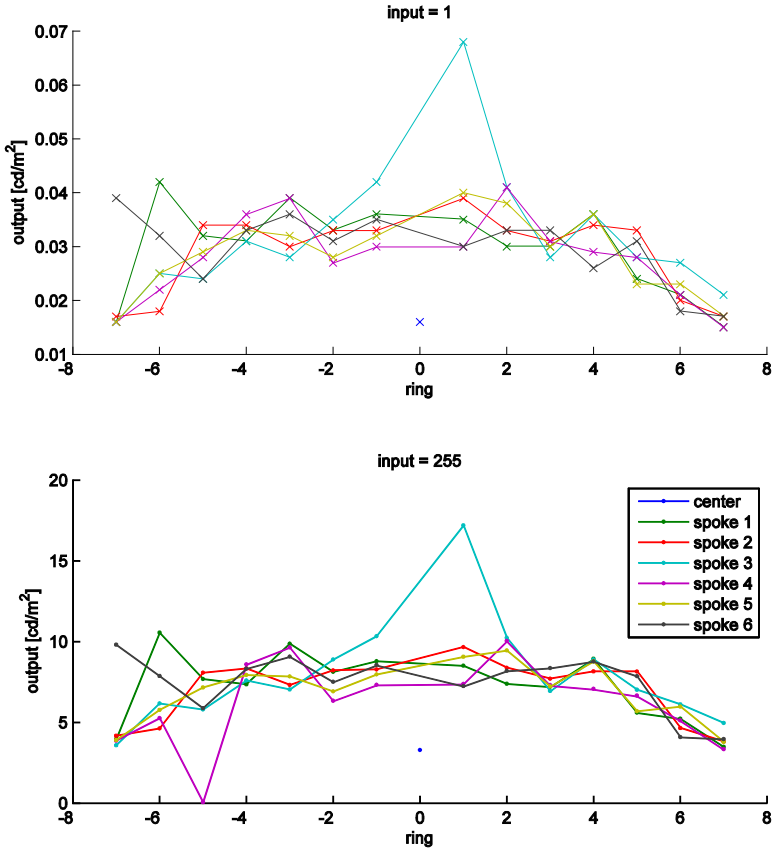

Figure 2. "LED Characteristics"of the LED-sky in the FART1 experimental room. Measured February 14th, 2008. Measurements were taken at a distance of 104 cm with the following settings of the lux-meter: non color-corrected, preset, continuous, absolute, slow response. The LEDs did not fill the entire measurement area, so these values are not absolute. Note that the center speaker can be red or green (measured in green condition), while the other speakers are always green.The spoke 3/ring 1 LED is an outlier.

2 Design of the Experiment and Programs

It is assumed that the reader has Matlab installed, including the *Neural Network* and *Statistics* toolboxes (\\multi.science.ru.nl\install\Matlab-R2007b) and the "*Auditory Toolbox*" (\\plus.science.ru.nl\mbaudit1\MATLAB_STUFF\auditory_toolbox). You will need an account and password to access the network.
It is also assumed that you have rudimentary knowledge about digital signal processing (for an extensive explanation, you can take a peak at http://www.dspguide.com/).

2.1 Human Localization Experiment

2.1.1 Your own folder on the Experiment-PC

By default, all experimental programs are found on the C:-directory of the experiment-PC, while all data- and stimulus files are stored on the D:-directory. You will need to create a folder for your own use on the D:-directory: *D:\HumanV1\User*, with *User* being your own account name. Within this folder you will usually need several sub-folders: Dat, Cfg, Exp, and Snd. The use for these folders

will be explained in the following sections.
You should only change files within your user-folder on the D:-directory! Never change anything on the C:-directory or on other people's folders on the D:-directory!

2.1.2 HumanV1

The main experimenting program is ***HumanV1***.exe (Figure 3) and is found in the directory *C:\HumanV1* on the experiment-PC.

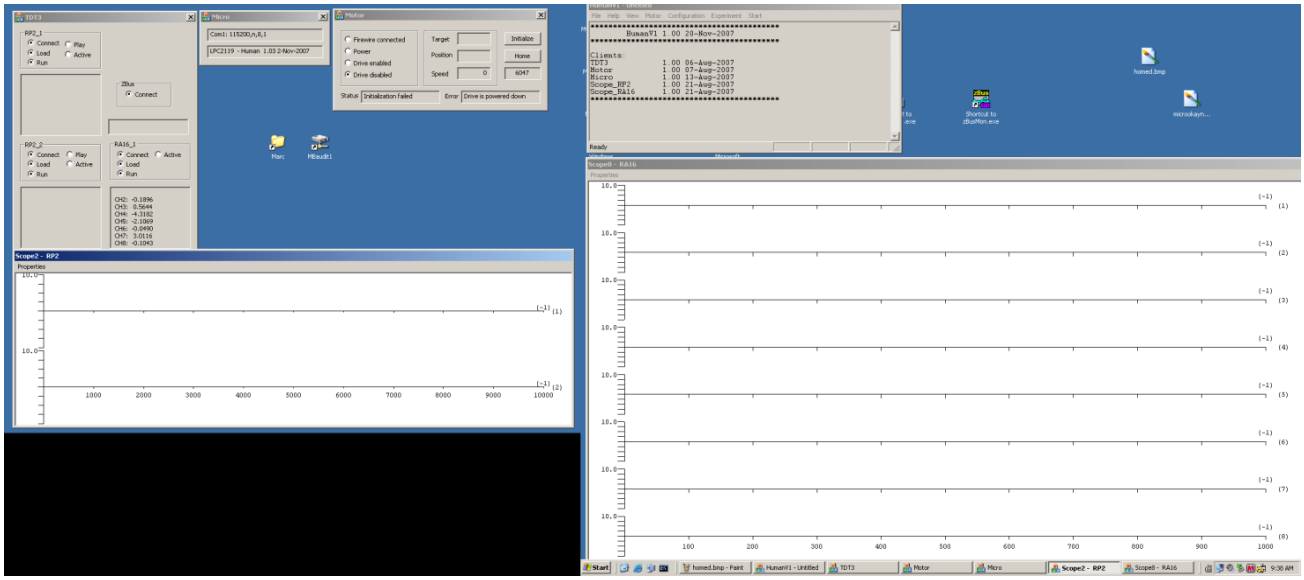

Figure 3. Overview of the HumanV1 user interface. The interface consists of 6 windows.

To run an experiment with ***HumanV1*** you need to create two input-files:

CFG Auxiliary file that contains the desired settings for the TDT Data Acquisition: which channels to use and at what sampling rate. This file is in ASCII-format. These files should by default reside in the directory *D:\HumanV1\User\Cfg*. Below, an example of such a file will be given.

EXP The experimentation file that contains a precisely-formatted ASCII list of all the stimulus locations, stimulus types, and the timing, as applied in the experiment. These files should by default be in directory *D:\HumanV1\User\Exp*. Below, an example of such a file is provided.

To start the program, double-click on the HumanV1 shortcut (Figure 4). This will create 6 windows as shown in Figure 3.

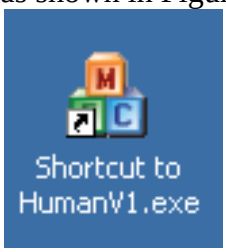

Figure 4. Shortcut to HumanV1.exe

The Main Window (Figure 5) allows for interactive control of the experiment, while the other 5 windows display information about the experimental set-up, such as the motor, microcontroller and RP2 and RA16. These 5 windows are crucial to check whether the experiment is running okay.

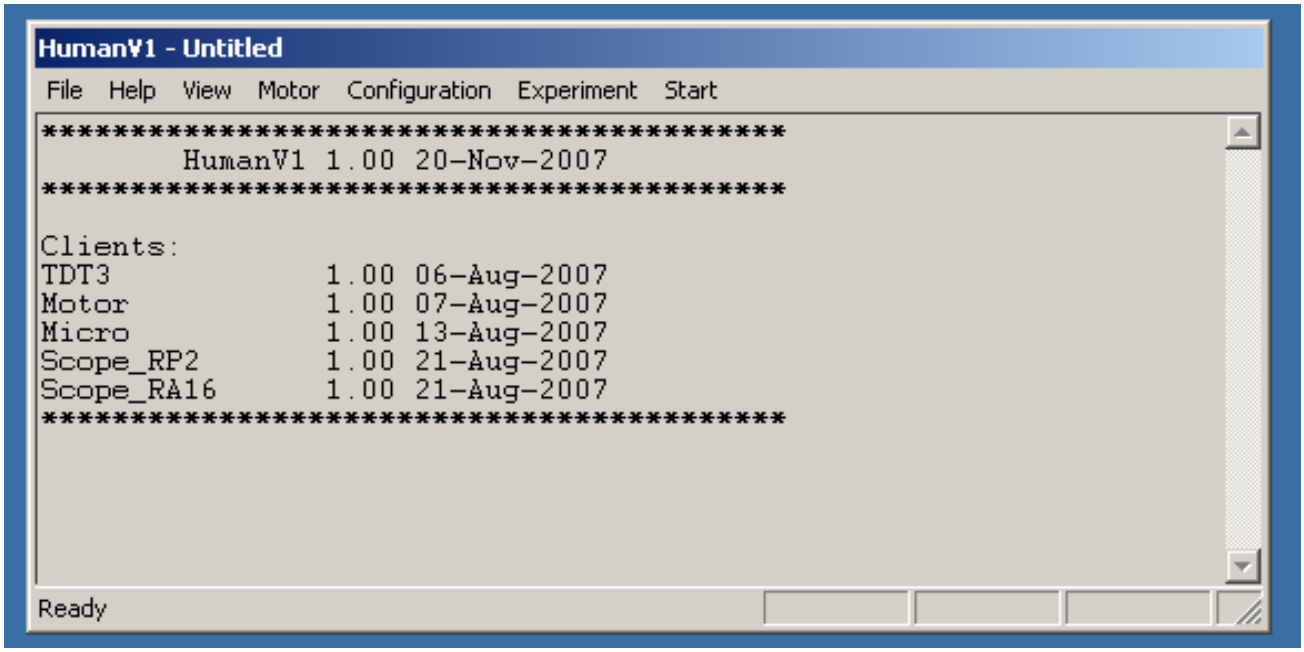

Figure 5. Main Window of HumanV1.

The top bar of the Main Window (Figure 5) is the title bar with the text: “Human V1 – Untitled”. The next bar is the task-bar and contains the following menus:

File:	SKIP
Help:	SKIP
View:	SKIP
Motor:	This menu allows you to move the hoop without intervention of an exp-file. The hoop can be positioned at 0 (home), 90 (right), 180 (back), 270 (left) and 360 (front again) deg. This function is never used during actual experiments, but it can be useful to center the hoop before or after an experiment.
Configuration:	This will start a pop-up menu from which you can choose a CFG-file (by default located in *D:\HumanV1\User\Cfg*).
Experiment:	This will start a pop-up menu from which you can choose an EXP-file (by default located in *D:\HumanV1\User\Exp*).
Start:	This will ask how to name the data-file that will be created during the experiment and start the experiment (after you have chosen a CFG- and EXP-file).

The window under the task-bar contains information about the experiment (which hardware is used, which cfg and exp-files are loaded, which trials are being played). **Trialnumber – past trials, trial 1/29 pending trial of 29 trials**

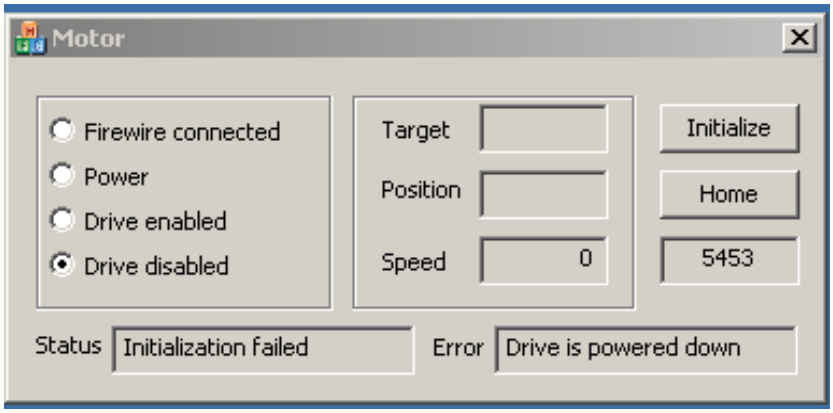

Figure 6. Motor Window

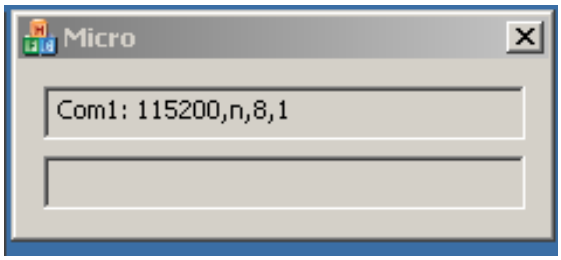

Figure 7. Microcontroller Window

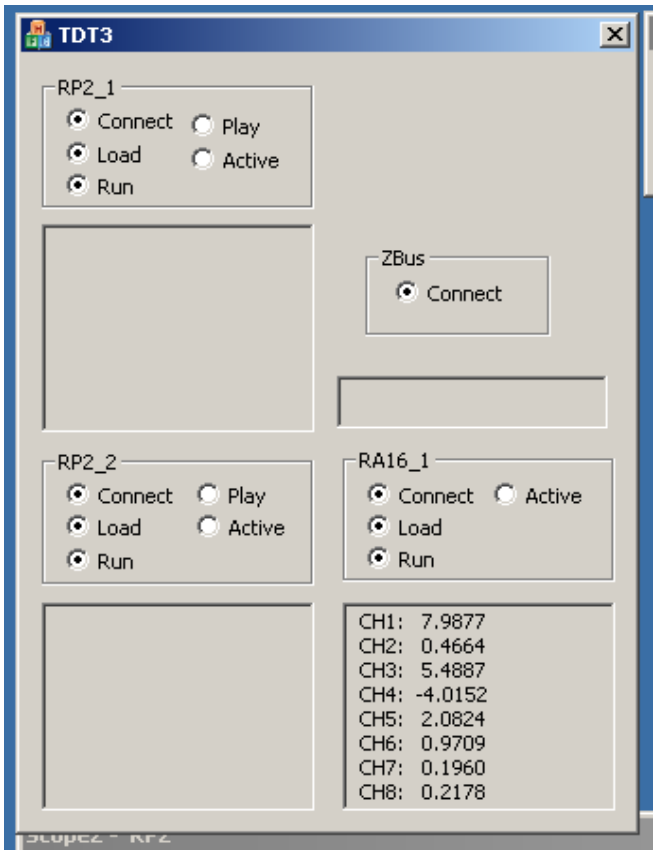

Figure 8. TDT Window.

Depending on the types of stimuli used, ***HumanV1*** will create one to four output-files, containing the data and the actual stimulus parameters:

DAT The experimental output file for head- and eye movement data (Remmel). In the data file, raw position signals are written (in alternating order, e.g. in case of acquisition of four channels (numbers 1-4):
ch1(0), ch2(0), ch3(0), ch4(0), ch1(1), ch2(1), ch3(1),..., ch1(n), ch2(n), ch3(n), ch4(n)

HRTF The output file for high-sampling rate data. There are a maximum of 2 channels, each corresponding to one of the two RP2s

CSV/LOG Comma-Separated-Value file and the LOG file (ASCII) containing the actual applied stimulus presentations. An example will follow below.

2.1.3 CFG Files

The CFG files define the desired Data Acquisition parameters for the TDT and computer board. These settings are copied into the output LOG-file. The successive non-commented lines (those that are not preceded by a %) in this file contain:

1. The directory where the log-files will be saved **!!!NO LONGER!!!**

2. The directory where the dat-files will be saved
3. The directory where the exp-files will be read from
4. The directory where the wav-files will be read from
5. The pathname of the first RP2 RPvdsEx circuitry
6. The pathname of the second RP2 RPvdsEx circuitry
7. The pathname of the RA16 RPvdsEx circuitry
8. The last lines contain the data-sampling configuration of the Analog-to-Digital channels (ADCs):
   a. Channel name (ADC1, ADC2,…ADCN)
   b. Lowpass cut-off frequency (for the digital filter): presently, this cannot be varied and is fixed at 250 Hz
   c. Sampling rate (in Hz) : at present this is fixed at 1017 Hz .
   d. The number of samples per trial. So the total trial acquisition duration for the channel is given by d/c seconds.
   e. Descriptive channel name (it is not necessary to provide one).

   A maximum of 8 channels can be defined.

An example:

```
%
% Configuration file: D:\HumanV1\User\CFG\Default.cfg
%

% Folders
LOGMAP      "D:\HumanV1\User\LOG"
DATMAP      "D:\HumanV1\User\DAT"
EXPMAP      "D:\HumanV1\User\EXP"
SNDMAP      "D:\HumanV1\User\SND"

% don't change the next two lines
RP2_1  "C:\\HumanV1\\RPvdsEx\\hrtf_2Channels_rp2.rco"
RP2_2  "C:\\HumanV1\\RPvdsEx\\hrtf_2Channels_rp2.rco"
RA16_1 "C:\\HumanV1\\RPvdsEx\\Remmel_8Channels_ra16_1.rco"

% Channel     | lp (Hz)     | Rate (Hz)   | samples (#) |
ADC1            250           1017          2000        HeadVertical
ADC2            250           1017          2000        HeadHorizontal
ADC3            250           1017          2000        HeadFrontal
ADC4            250           1017          2000        Button
```

You can create a new cfg-file by simply typing one in a text-editor like Notepad, or by typing:

```
>> gencfg
```

in the Matlab command window (Figure 9). By means of this GUI , you can easily specify the number of data acquisition channels you wish to use, and their configurations.

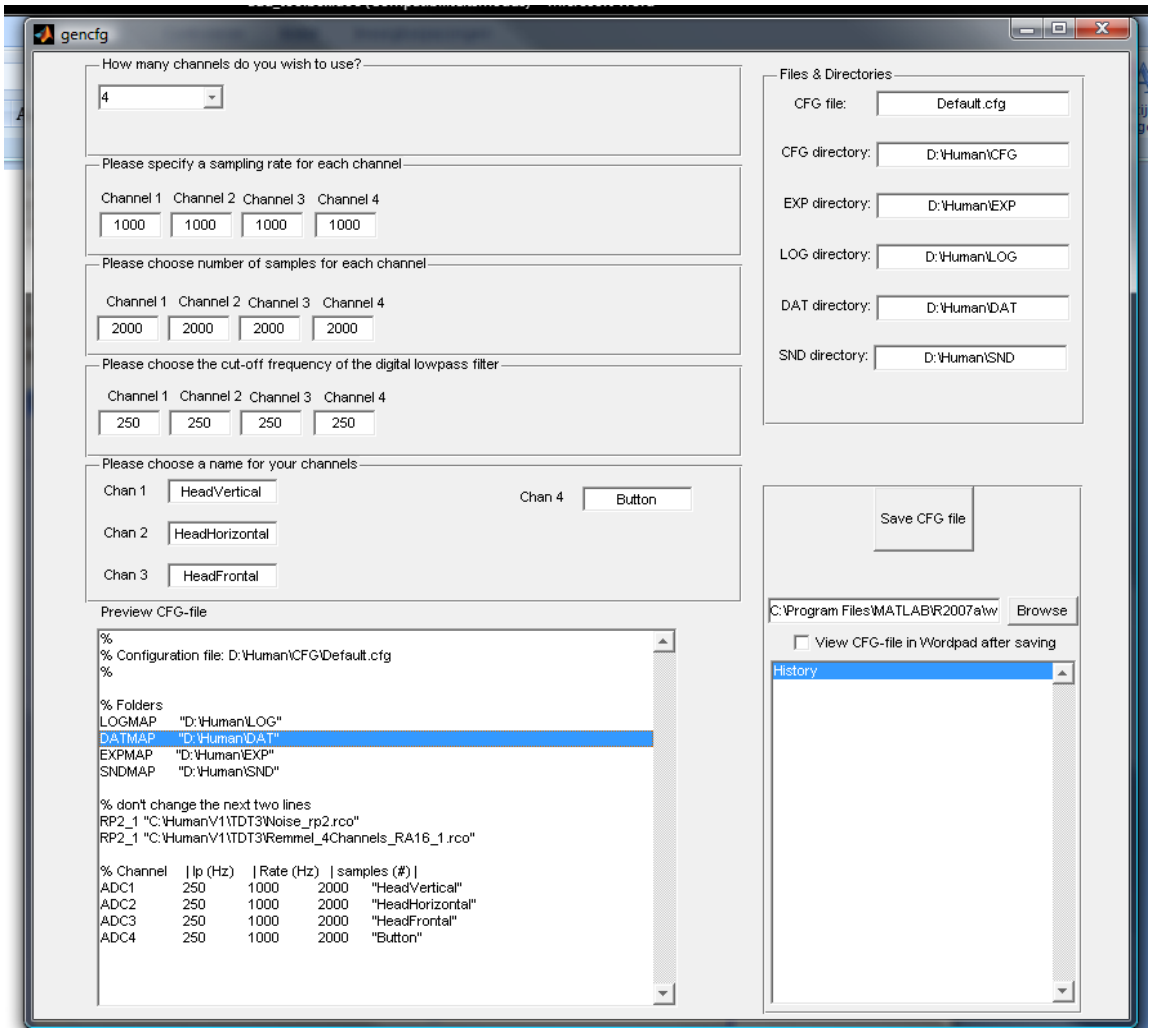

Figure 9. The Graphical User Interface for creating a configuration CFG file.

2.1.4 EXP files

The EXP files specify all stimuli that are going to be presented in the experiment. The file consists of two parts: the HEADER and the BULK. In the HEADER, several overall parameters are specified:

ITI: Inter Trial Interval, specify the minimum and maximum gap (in msec) between each trial. A random interval in the range [min,max] is picked out, separately for each trial.
Trials: The maximum number of trials in a session
Repeats: The number of times the trials should be repeated
Random: Randomization of trials, 0=no, 1=per repetition, 2=across all trials and repetitions
Motor: Movement of the robot, y = yes, n = no.

The BULK defines the different trials in the experiment. Each "==>" followed by several lines corresponds to a specific trial, where each line corresponds to 1 stimulus or event. Each stimulus is specifed by its own parameters, which will be described below.

MOD: Stimulus Modality (Led, Sky, Trg0, Snd, Acq)
X: Stimulus Eccentricity (deg), re center

Y:	Stimulus Elevation (#)
ID edge:	Trigger Edge (down, release)
INT bit:	Stimulus Intensity
ON event:	Stimulus Onset Event
ON time:	Stimulus Onset Time re to Stimulus Onset Event (ms)
OFF event:	Stimulus Offset Event
OFF time:	Stimulus Offset Time re to Stimulus Offset Event (ms)
EVENT:	Trigger Event (>0 = triggered event)

2.1.4.1 Stimulus Modality:

MOD defines the modality of the stimuli.

Led: Visual Stimulus. Corresponding to one of the 60 LEDs on the FART.
Sky: Visual Stimulus. Corresponding to one of the N LEDs on the LED-sky.
Snd1: Auditory Stimulus number 1. A sound will be presented through one of the speaker(s).
Snd2: Auditory Stimulus number 2. A second sound can be played through another speaker.

MOD also defines the parameters of 4 non-stimulus events:

Acq: Data Acquisition for sampling rates of 1000 Hz. These are usually used for eye and head position data. Acq is an event for all channels defined in the CFG-file.
Inp1: Data Acquisition for a single channel with a sampling rate of about 50000 Hz. These are used for sounds/hrtf measurements.
Inp2: As Inp1 but adds another channel.
Trg0: Trigger.

Note that the stimuli and events do not have the same number of parameters (e.g. X and Y parameters are only used by Led, Sky and Snd).

2.1.4.2 Stimulus Eccentricities and Elevations

Led: These are expressed in FART coordinates, where the center of the sphere is taken as $[R,\varepsilon]=[0 \text{ deg},12]$. Eccentricity R is defined by the position of the hoop (front = 0 deg, left is -90 deg, right is 90 deg, etc). ε is defined by the LED location on the hoop. Elevations are restricted to the discrete values on the hoop. $\varepsilon \in [1,2,\ldots,30,31,101,102,..,129]$. LEDs 1-29 are located in the front-hoop, 101-129 are in the rear-hoop. 30-31 are two extra LEDs at eye-height (0 deg), at left and right of the hoop. The LEDs are separated by 5 deg, the front lowest LED (1) is located at -55 deg, the rear lowest LED (101) is located at -57.5 deg.

Snd1/2: Are also given in the same coordinates as Led.

Sky: The X coordinate corresponds to the spoke number (1-12), Y corresponds to the ring/LED number on the spoke (1-7). The fixation LED has X coordinate 0, and no Y coordinate.

2.1.4.3 Stimulus/Event Identity

Snd1/2: The identity parameter determines the *Stimulus File Number* that the ***HumanV1*** program will play (snd*XXX*.wav, with *XXX* indicating the file number)

Trg0: The identity parameter determines the action which will trigger an event. Button

Press = 1, Button Release = 2.

Sky: Sky does not use this parameter except for the center fixation led (X = 0). The identity parameter for this LED codes color: red = 1, Green = 2.

Led, Acq, Inp1 and Inp2 don't have identity parameters.

2.1.4.4 Stimulus Intensity

Led: The intensity of a LED can vary from 1 (low) to 255 (high).

Sky: The intensity of a Sky LED can vary from 1 (low) to 255 (high).

Snd1,2: Sound intensity is implemented in a range from 0-100 in ARBITRARY UNITS; actual sound level in dBA varies per stimulus (see Figure 10; for a 150 ms 0.2-20kHz sound with a 5 ms on- and offset slope will range from 30-70 dBA for exp levels 20-60; the actual sound level in dBA should be measured).

Trg0: The micro-controller has 8-bits digital input (TTL), inputs 5..8 have a debounce circuitry. Default input for the button is 5.

Acq, Inp1, and Inp2 don't have intensity parameters.

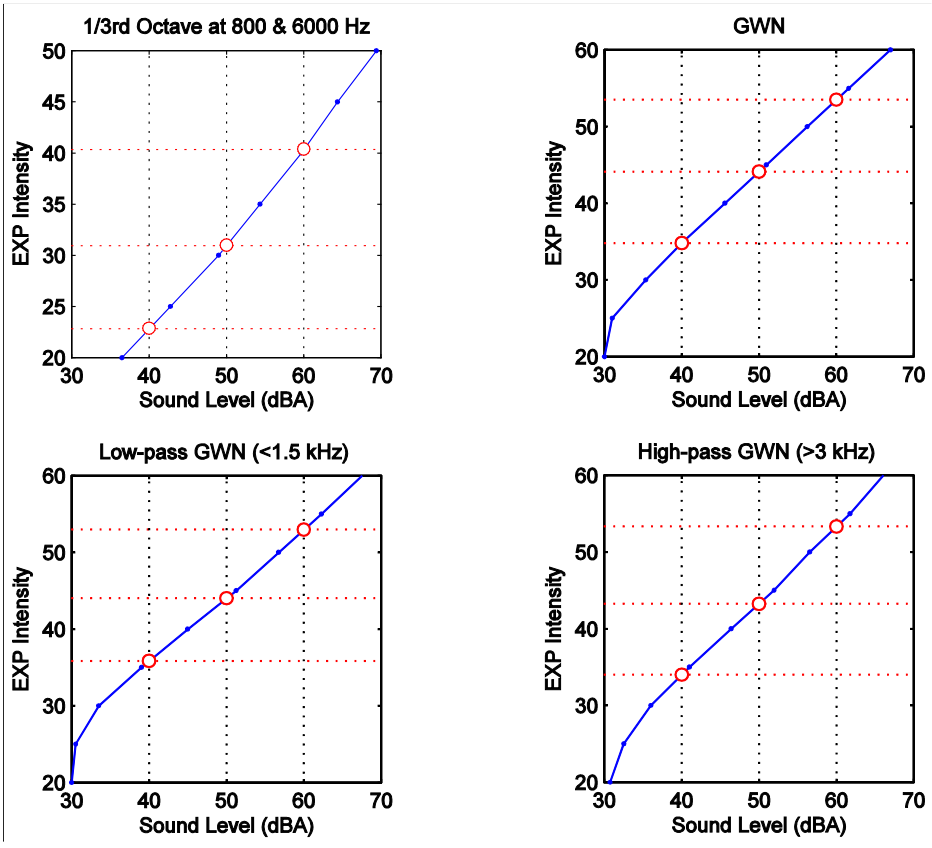

Figure 10. Relation between sound level (dBA) and intensity as defined in exp-file for various

stimuli. Sounds (10s duration) were delivered from speaker 12 (straight ahead), and measured with the Bruel & Kjaer.

2.1.4.5 Timing

On Time and *Off Time* are given in msec, relative to the Event specified by *On Event* and *Off Event* 0 is defined as the start of the trial. You can define other Events, by supplying a Trg0-line.

Led: Uses both Onset and Offset Parameters

Sky: As Led.

Snd1: Has Onset Parameters. Offset is determined by the length of the sound file NOT by the *Off Time* and *Event* in the EXP-file.
Snd2: As Snd1.
If Snd1 and Snd2 have the same Onset, a trigger pulse will ensure that they start at the same time. Optionally, you could add the extra shift parameter for Snd2. If you do so, then pad your wav-files with zeros by at least the number of samples in the shift.

Acq: will start at *On Time* msec after *On Event*. Offset does not need to be specified. If left out, Acq will have a duration as specified in the CFG-file. Otherwise, you will only need to specify the duration of the acquisition (thus, no *Off Time* and *Off Event*).

Trg0: Uses only Onset Parameters.

Inp1 (and Inp2) doesn't have Timing parameters (yet). They start at the same time as Snd1 (Snd2), and end 100 samples later.

2.1.4.6 Limitations

There are some limitations to the experimental configurations:

1) Acq channels can contain a maximum of of 10^5 samples each
2) Snd1 and Snd2 stimuli can contain a maximum of 10^6 samples
3) Inp1 and Inp2, same as Snd1 and Snd2
4) There is a maximum of 20 stimuli/events (lines) per trial (➔)
5) Each exp-file has a maximum of 2000 stimuli/events

2.1.4.7 Example EXP file

Example stimulus/event lines:

```
% MOD X     Y     ID    INT   On    On    Off   Off   Event
%                 edg   bit   Event Time  Event Time
```

A green, low-intensity fixation led on the sky starting 0 ms after onset trial, and ending 200 msec later:

```
Sky   0           2     50    0     0     0     200
```

High-intensity led on the sky (spoke 1, led nr 3), starting 0 ms after onset trial, and ending 200 msec later:

```
Sky   1     3           255   0     0     0     200
```

Low-intensity led on the hoop at 90 deg to right and eye-height (nr 12) starting at 0 ms after onset trial, and ending 200 msec later:

```
Led   90    12          3     0     0     0     200
```

Play a half-maximum intense sound (wav-file nr 001) at 40 deg to the left and 20 deg down (speaker nr 8), starting 250 msec after onset trial (and ending as determined by the length of the wav-file):

```
Snd1  -40   8     001   50    0     250
```

Play a second sound (wav-file nr 003) at 20 deg to the right and 5 deg up (speaker nr 13), starting 250 msec after onset trial +10 samples (= +10 samples after onset Snd1):

```
Snd2  20    13    003   70    0     250   10
```

Start Data Acquisition 100 msec after start of trial (and end as determined by CFG-file):

```
Acq                           0     300
```

Start Data Acquisition 100 msec after start of trial, and end 200 msec later:

```
Acq                           0     300   200
```

Incorporate an event triggered by a button release:

```
Trg0              1     5     0     0                 1
```

Read from high-frequency Data Acquisition channels, Inp1 and Inp2 (without any inputs):

```
Inp1
Inp2
```

Short example of two trials in an EXP-file, including the HEADER. Each trial starts with a low-intensity LED straight ahead. After button press, this fixation LED is extinguished and a second LED at a different position is turned on. A second button press extinguishes this LED, and will start data acquisition and play the sound.

```
%
%% Experiment: C:\Human\DAT
%
ITI         0     0
Trials      392
Repeats     1
Random      0     % 0=no, 1=per set, 2=all trials
Motor       n

% MOD X     Y     ID    INT   On    On    Off   Off   Event
%                 edg   bit   Event Time  Event Time

==>
LED   0     12          1     0     0     1     10
Trg0              1     5     0     0                 1
LED   0     9           1     1     100   2     1
Trg0              1     5     1     0                 2
Acq                       2     1
Snd1  0     27    820   74    2     201

==>
LED   0     9           1     0     0     1     10
Trg0              1     5     0     0                 1
LED   0     6           1     1     100   2     1
Trg0              1     5     1     0                 2
Acq                       2     1
Snd1  0     25    220   72    2     201
```

2.1.4.8 Creating an EXP-file

You can create a new exp-file by simply typing one in a text-editor like Notepad. You could also try

to create an m-file in MATLAB to do it for you, as done in:

```
>> genexperiment
```

This m-file will be explained later.

2.1.5 DAT files

The DAT-files contain the data: eye position, head position, button press, sound amplitude, etc. The data are stored as 32-bits floating point numbers ("float" in Matlab). To read a DAT-file into MATLAB:

```
>> [Dat]=loaddat(<dat-filename>,<number of channels>,<number of samples>);
```

In a later section, the analysis of the data will be explained.

2.1.6 LOG/CSV files

The CSVfile contains all the relevant parameters of the experiment. The first line provides general information about the experiment.

1) 0
2) Maximum number of trials per repetition
3) Number of repetitions
4) Number of trials (2)*3))
5) Inter Trial Interval start
6) Inter Trial Interval end
7) Randomization Type
8) Number of (Acq) channels

The following lines provide data acquisition channel information:

1) 0
2) Channel Number
3) Channel Number
4) Low-pass cut-off frequency
5) Desired Sampling rate
6) Number of samples

Each line corresponds to one channel.

Hereafter, the different stimuli and events are given line by line (in much the same way as in the EXP file, albeit in the actual order of stimulus presentation):

1) Trial number
2) Stimulus number in trial
3) Desired Inter Trial Interval
4) Actual Inter Trial Interval (can deviate from 3) when e.g. disk is too busy)
5) Modality of stimulus
6) Hoop location (degrees, -180..180) / Spoke (1-12)
7) Speaker/Led location (number)
8) Stimulus onset relative to trial onset
9) Stimulus offset relative to trial onset
10) Stimulus intensity (for LED: 0 = lowest ... 7 = highest, for SND: 0 = lowest ... 100 = higest)
11) Stimulus attribute (For LED: 0 = red, 1 = green; for SND: XXX (fragment of wav-name -> sndXXX.wav))
12) Bit (Micro-controller Trg0: 5; For SND: XXX (fragment of wav-name-> sndXXX.wav))
13) line number in EXP-file (which deviates from 1) when trials are randomized according to the EXP-file)

Onset and offset timings will in general deviate slightly from the intended timings in the EXP-file, due to small random time delays in the processing. The actual timings in the CSV-file, however, are exact. From this, one is able to reconstruct the actual stimulus configuration, e.g. at which time the stimuli were triggered.
You can read the data from a CSV-file into Matlab:

```
>> [expinfo,chaninfo,log]=readcsv(<csv-filename>);
```

2.1.6.1 Example CSV file

```
0;56;1;56;100;100;0;8
 0;1;1;250;1000;100
 0;2;2;250;1000;100
 0;3;3;250;1000;100
 0;4;4;250;1000;100
 0;5;5;250;1000;100
 0;6;6;250;1000;100
 0;7;7;250;1000;100
 0;8;8;250;1000;100
  1; 1; 100; 100;Trg0; NaN; NaN; -364;    0; NaN; NaN;   2;   1
  1; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   1
  1; 3; 100; 100; Led;  90; 101;-2633;  100;   7;   1; NaN;   1
  2; 1; 100; 100;Trg0; NaN; NaN; -273;    0; NaN; NaN;   2;   2
  2; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   2
  2; 3; 100; 100; Led; -90;   4;-1245;  100;   7;   1; NaN;   2
  3; 1; 100; 100;Trg0; NaN; NaN; -208;    0; NaN; NaN;   2;   3
  3; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   3
  3; 3; 100; 100; Led; -90;   8;-2231;  100;   7;   1; NaN;   3
  4; 1; 100; 100;Trg0; NaN; NaN; -168;    0; NaN; NaN;   2;   4
  4; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   4
  4; 3; 100; 100; Led; -90;  12;-1727;  100;   7;   1; NaN;   4
  5; 1; 100; 100;Trg0; NaN; NaN; -188;    0; NaN; NaN;   2;   5
  5; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   5
  5; 3; 100; 100; Led; -90;  16;-1798;  100;   7;   1; NaN;   5
  6; 1; 100; 100;Trg0; NaN; NaN; -155;    0; NaN; NaN;   2;   6
  6; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   6
  6; 3; 100; 100; Led; -90;  20;-2068;  100;   7;   1; NaN;   6
  7; 1; 100; 100;Trg0; NaN; NaN; -169;    0; NaN; NaN;   2;   7
  7; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   7
  7; 3; 100; 100; Led; -90;  24;-1745;  100;   7;   1; NaN;   7
  8; 1; 100; 100;Trg0; NaN; NaN; -162;    0; NaN; NaN;   2;   8
  8; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   8
  8; 3; 100; 100; Led; -90;  28;-1861;  100;   7;   1; NaN;   8
  9; 1; 100; 100;Trg0; NaN; NaN; -163;    0; NaN; NaN;   2;   9
  9; 2; 100; 100; Acq; NaN; NaN;    0;  800; NaN; NaN;1017;   9
  9; 3; 100; 100; Led; -80;  12;-2214;  100;   7;   1; NaN;   9
```

3 Creating an experiment

3.1 Designing Auditory Stimuli in MATLAB

Although a large number of acoustic stimuli can be found on the experimental PC, it is often desirable to design your own. A number of MATLAB routines have been developed to generate single channel wav-files. Note that the sampling rate for DA-conversion is always 48828.125 kHz, so that the inter-sampling distance is 20.48 µsec.
The following useful MATLAB routines may be used for the most frequently used stimulus types:

gengwn - generate Gaussian White Noise (by defining a randomly distributed time signal)
gengwnflat - generate Gaussian White Noise (by defining a flat magnitude and random phase in the frequency domain)
gensweep - generate a frequency sweep with a flat spectrum
gentone - generate a pure tone
(the above functions generate MATLAB arrays)

psd - MATLAB routine to plot the power spectral density

writewav - writes the stimulus as a wav file for the DA converter
wavread - MATLAB routine to read an existing wav-file in MATLAB format

Some examples:
Let's generate a GWN stimulus. Stimulus 3.0 sec, bandwidth from 0.5 to 20 kHz, with on- and offset envelopes.
In Matlab:

```
>> Freq = 48828.125;
>> Nsamples = round(Freq*3)   % approx. 48828.125*3 samples
>> Noise = gengwn(Nsamples);  % Default envelope (250 pnts) and filter
>>                            % order (100)
>>                            % noise is lp-filtered at 22.5 kHz
>>                            % and hp-filtered at 300 Hz
>> psd(Noise);
>>
```

Frequencies below 300 Hz are never useful, and produce distortions on speakers, so they should be filtered out with a highpass-filter. This is already done by default in `gengwn`.
The GWN stimulus is a well-localizable sound eliciting all binaural difference (ITD and ILD) cues and spectral cues. Sometimes, it is desirable to use lowpass- or highpass-filtered noise, to separate the effects of ITD and ILD. These sounds are created by generating a GWN as above, and filtering it with the functions lowpassnoise and highpassnoise:

```
>> HP = highpassnoise(Noise);            % noise is hp-filtered at 3 kHz
>> psd(HP);
>>

>> LP = lowpassnoise(Noise);             % noise is lp-filtered at 3 kHz
>> psd(LP);
>>
```

In the FART1-setup, speakers will have on onset ramp produced by the TDT-system to ramp to a voltage on the speakers. You will have to take this into account, by putting zeros in front of the

stimuli (of about 20 ms worth = 20*48.8828125 = 978 samples):

```
>>Nzero = zeros(1,978);       %This will create a vector with 978 zeros
>>Noise = [Nzero Noise];      %This "prepends" the zerovector to the Noise
>>HP = [Nzero HP];
>>LP = [Nzero LP];
```

After this, you should save the matrix to a wav-file, that can be played with the TDT-system:

```
>> writewav(Noise,'snd001.wav');
>> writewav(HP,'snd002.wav');
>> writewav(LP,'snd003.wav');
```

To change default parameters in these functions, you can provide additional input parameters. To know which input paramaters can be changed, type:

```
>> help <mfile>
```

e.g.

```
>> help gengwn
 Generate Gaussian White Noise Stimulus

  GWN = GENGWN (N, NEnvelope, order, Fc, Fn, Fh)

  Generate Gaussian White Noise Stimulus, with
  N         - number of samples
  NEnvelope - number of samples in ramping envelope
  Order     - order of filter
  Fc        - Cut-off Frequency
  Fn        - Nyquist Frequency

  For example:
    N = 7500;
    stm = gengwn(N,250,100,20000,25000)
    fname = 'BB.wav';
    writewav(stm,fname);


  will generate a broad-band noise between 20 (default) and 20 kHz with
  duration 150 msec (7500 samples / (25000 samples/sec*2) *1000). The
  on- and offset ramp each contain 250 samples = 250/50000 sec = 5 msec.
  This noise will be stored in the WAV-file 'BB.wav'.

  See also writewav, lowpassnoise, highpassnoise
```

3.3 Generating a "random" experiment (EXP-file) in MATLAB

The major advantage of using a Matlab-generated exp-file, as opposed to generating it yourself in a text-editor, is that you can easily randomize stimulus parameters to ensure a maximum amount of uncertainty for the subject. For example, you might want to randomize the onset times of your sounds in order to reduce prediction by the subject. This randomization is difficult for you (humans are notoriously bad random number generators), but the computer can do it easily.

The major disadvantage is that you cannot have MATLAB forsee all possible experimental configurations.

This section will try to outline some useful MATLAB-procedures in creating an exp-file.

For a maximum amount of uncertainty for the subject, yet at the same time a homogeneous distribution of targets across the stimulus range, the mfile `getloc` can be used as follows:

```
>> nrgrid = 4;          % sqrt(nr of grids)
>> nrloc = 3;           % nr of locations per grid
```

```
>> [theta,phi] = getloc(nrgrid,nrloc)
```

This function first determines all possible locations that are provided by the speakers/LEDS of the FART-hoop. It then divides these locations in nrgrid^2 grids (in this example 4*4=16). For each of these grids, nrloc-1 random locations are chosen (in this case 2), and the last location of each grid is obtained such that all locations in a grid average out to the center of the grid (**Error! Reference source not found.**).

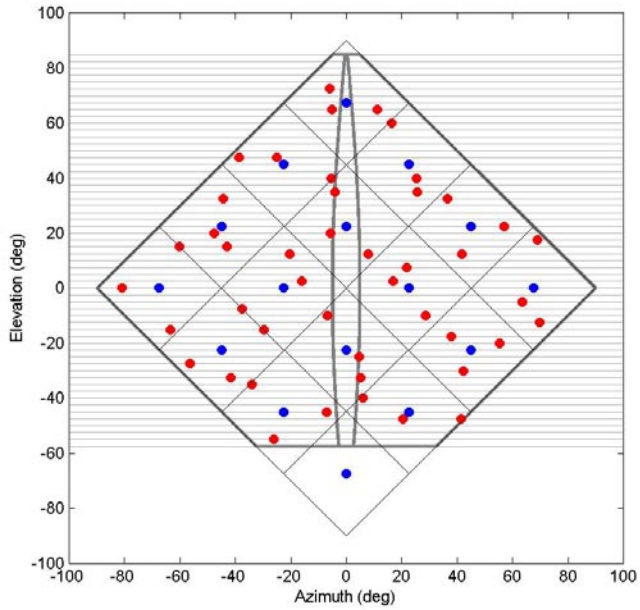

Figure 11. Random locations obtained with getloc. Blue denotes the mean target location of each grid, red denotes the actual (pseudorandom) locations in each grid. Horizontal grey lines are the possible elevations of the speakers/LEDs.

Each location is described by two location parameters: theta and phi. Theta is the angle of the motor (-180 to 180 deg), while phi is the number of the speaker/LED (1-29,101-129). If you want to convert these FART coordinates to the 2D double-polar coordinates of azimuth and elevation, use `fart2azel`:

```
>> [azimuth,elevation]=fart2azel(theta,phi);
```

To write these locations to an exp-file, use `writeloc`:

```
>>writeloc('myexp.exp',theta,phi,'snd');
```

Getloc also ensures that any auditory or visual stimulus on the hoop will not hide the straight-ahead fixation-LED on the sky.

4 Protocol for Human Experiments

Design an Experiment
Create a Sound stimulus (Chapter 3)
Create an Exp and CFG file (Chapter 3)
Possibly Check for actual Sound Intensity with Bruel & Kjaer
Start the TDT and Microcontroller (Chapter 1)
Start the experiment-PC (Chapter 1) and put all experiment files on your d:\HumanV1\User directory (Chapter 2)
Start the motor (Chapter 1)
Start HumanV1 (Chapter 2)
Check whether everything is a-okay
Home Motor
Start inzwaaien.exp
Never walk in the room when the motor is on (red light shines).
For the first time: test Experiment without subject
For the first time: test Experiment with experimenter as subject
If everything is okay, get subject
Test subject:
- Coil/DMI
- Check gains/offsets Remmel
- Calibration – calgrid.exp
- CFG
- EXP
- Localization
- CFG
- EXP
- Start (with convention XX-SS-YYYY-MM-DD)

Yell to subject (‘Okay!’, ‘Goed zo!’, ‘Prima!’, ‘Vooral zo doorgaan!’) every 50 trials and pause every 200 trials to keep subject awake and fit (this varies per subject).
Turn off motor (red light off).
Subject may leave room.
Back-up data on network (mbaudit1).
Analyze data on your PC (Chapters 5-7).

5 Calibration of Human Data

This chapter describes the steps needed to transform the AD-numbers in the raw data files (*.DAT) into degrees-calibrated position signals.

5.1 Eye Position Calibration

To calibrate Eye Position signals, the relation between the actual orientation of the coil and the eye and the associated AD numbers from the TDT AD2-module has to be determined. Therefore, the subject has to fixate (FART- or Sky-)LEDs with known positions in the set-up. To indicate that he is doing this, the subject should press a button, which starts the data acquisition.

An example trial in a calibration experiment:

```
% MOD X     Y     ID    INT   On    On    Off   Off   Event
%                 edg   bit   Event Time  Event Time
==>
Trg0              1     5     0     0                 1
Acq                           1     0     100
LED   90    12          255   0     0     1     100
```

It is important to start data acquisition after the subject has properly fixated the LED (which he can indicate by a button press). The subject then has to keep fixation for as long as data is acquired.

5.2 Neural Network

To learn the relation between eye position and AD data, a feedforward neural network, implemented in MATLAB's neural network toolbox, is used (5-hidden units-layer back-propagation). In what follows, the most important steps are described by an example calibration-experiment, named XX-SS-YYYY-MM-DD-0000.

You will need to call the MATLAB-GUI traincal on the calibration experiment:

```
>> traincal('XX-SS-YYYY-MM-DD-0000')
```

This GUI wil automatically train the network on the AD-traces in the DAT-file and the target position in the CSV-file. After training you wil see the GUI appear as in Figure 12.

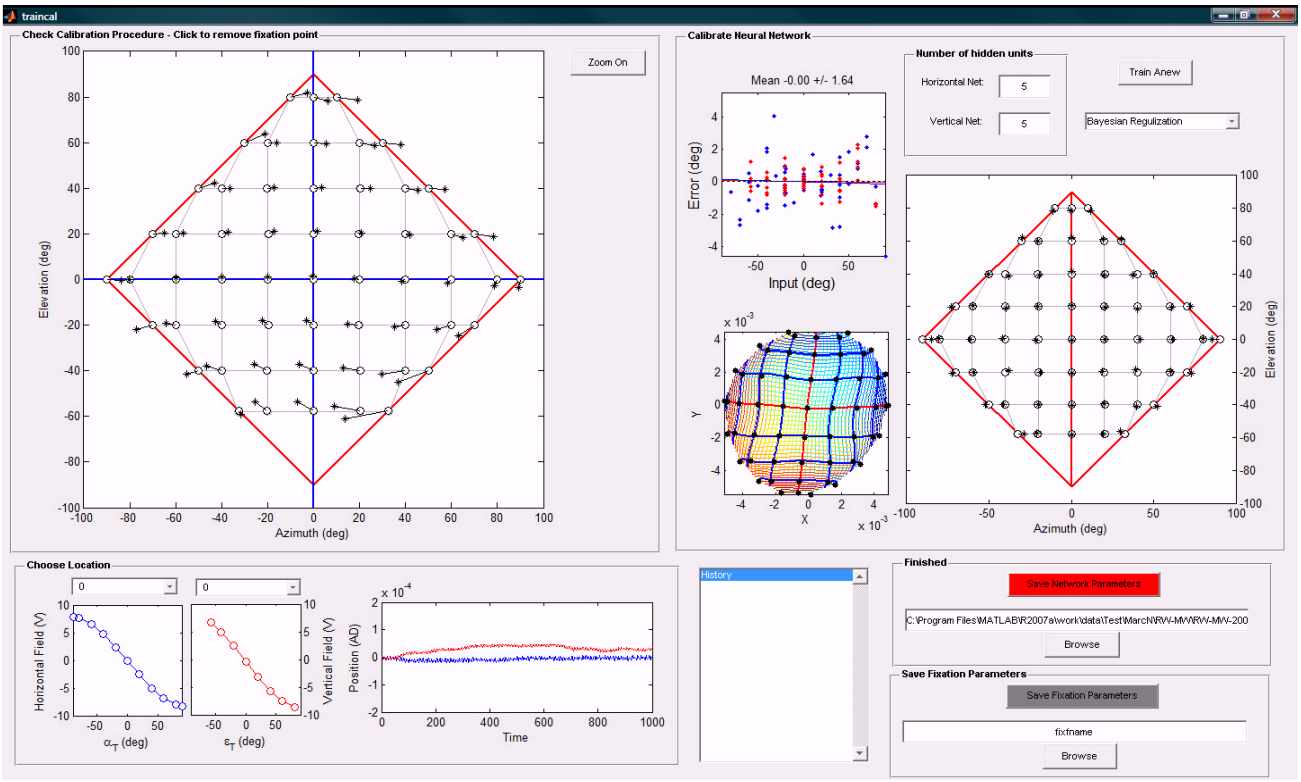

Figure 12 Graphical User Interface TRAINCAL for training the calibration neural network.

You have to check for outliers in the fixation experiment in Figure 12A, where the end positions are displayed together with the corresponding target positions. By simply clicking you can remove any undesirable outlier. In the example, the subject seems to have fixated well. There are some deviations between the target and the fixation position, but they are systematic (due to non-linearities in the magnetic fields), and they can be easily taken care of by the network. In Figure 12B, you can specify the azimuth and elevation components, so you can observe the time trace of one fixation.

If you have removed any outliers, you will need to re-train the network. This is done in Figure 12C by pressing the “Train Anew”-button. The neural networks are trained by applying the Levenberg-Macquardt gradient descent method with Bayesian regulization (see Neural Network Toolbox). Two networks are trained separately: one network for the horizontal component of eye position and one for the vertical position component. The actual target positions serve as the teacher signal for the network. The end result of the learning procedure, the weights of the network, are stored in a so-called NET-file by pressing the “Save network parameters”-button.
The performance of the trained network is shown in the three figure axes in C. If you do not like the performance, you can repeatedly try to train the network again (“Train Anew”-button) and save the desired network (“Save network parameters”-button). The average errors for horizontal and vertical fixations (C-I) should be virtually zero, and there should be no systematic deviations. The typical standard deviation should be in the range of 1.8 deg, or less. Furthermore, the iso-azimuth and iso-elevation lines in figure C-II of the X and Y magnetic fields should not contain too many strange warps (which indicate overfitting), and the target and calibrated fixation positions in C-III should not deviate systematically.

5.3 Calibration of the raw data

With the function >>calibrate all raw DAT-files can be calibrated in the current directory.

5.4 Low-pass filtering of the data

After the calibration, you will have obtained several hv-files. Since the data will without a doubt contain noise, it might be advisable to low-pass filter the data in the files before any further analysis is initiated (depending on the noise level). One possibility of doing this is by using the function *hvfilt*:

```
>> hvfilt;
```

A low-pass filter is applied to the raw hv-data, contained within the hv-files in the current directory. You can also supply *hvfilt* with the hv-files you want to filter, i.e.:

```
>> hvfilt(['XX-SS-YYYY-MM-DD-0000';' XX-SS-YYYY-MM-DD-0001']);
```

By default the original hv-files will be overwritten. To prevent that, you can supply output-files, i.e.:

```
>> hvfilt(['XX-SS-YYYY-MM-DD-0000'],['XX-SS-YYYY-MM-DD-0000f']);
```

Filtering with this function is achieved by using a 50-point digital low-pass FIR filter, with a cut-off frequency of 80 Hz. You can supply an additional parameter to the function, to change this frequency, i.e.:

```
>> hvfilt(['XX-SS-YYYY-MM-DD-0000'],['XX-SS-YYYY-MM-DD-0000f'],90);
```

6 Saccade Analysis

6.1 Saccade Detection

When calibration of the raw data files (see previous chapter) has been finished, saccades can be Detected. A saccade is defined by an onset and an offset moment-marking within the trial. These markings are stored in a so-called SAC-file, which is generated by a saccade detection function in MATLAB called *ultradet* (alternatively you can use *veldet*). By using this SAC-file, later analysis may concentrate on the right chunks of data in the original data-file for extracting the relevant parameters.
How *ultradet* works is briefly described in Appendix D. The current section describes how to use the program for detecting saccades from the different experiment types. Again, the procedure is illustrated by the example experiment XX-SS-YYYY-MM-DD-0001.

Eye Movements

The aim is to indicate the on- and onsets of all saccades in file XX-SS-YYYY-MM-DD-0001.hv. Therefore, type:

```
>> ultradet('XX-SS-YYYY-MM-DD-0001.hv');
```

in the data directory where the hv-file is stored. If you are in a different directory (or if you haven't supplied *ultradet* with a filename), a window will pop up and ask you to search for a file you want to have analyzed. The computer will then on the basis of the (filtered) eye velocity profiles determine the start and ending of saccades. In general, the program will detect the saccades correctly, but the experimenter may change the markings interactively, if needed. The detection parameters needed by the algorithm are specified in the so-called det-structure. Ultradet will by default choose the (near-optimal) settings for the eye detection saccades by itself, but you may alter them by supplying your own `det`-structure to *ultradet* (See Appendix D).

Eye-Head Movements

The detection of gaze (combined eye-head) movements requires three steps, because one may identify three types of saccades (Gaze saccades, Eye-in-head saccades, Head saccades). Each type should be detected separately.

6.2 Saccade Parameters

The extraction of the different saccade parameters is again done in MATLAB, by using the

CSV- and SAC-files. The MATLAB functions are available both for Windows and Unix environments_ The backbone analysis function that is always useful is the

program SACTOMAT and is evoked as follows_

□□sactomat_DataFile_ Nchan_ HVchan_ SacFile__

The output of this function is a MAT _le in which two matrices are stored_ namely a Sac matrix

that contains the saccade parameters_ and a Stim matrix_ in which stimulus parameters are

included_ These matrices will be further explicited in the next chapter

```
>>hvtomat
>>sac2mat
```

6.3 Cleaning the Data Directory

After calibrating the raw data files, saccade detection, and computation of the saccade parameters, there will be a large number of different files in the data directory. Not all these files are equally useful. When everything worked out fine, all files needed for further analysis of the results are compressed in three ZIP files:

- XX-SS-YYYY-MM-DDdat.zip
- XX-SS-YYYY-MM-DDsac.zip
- XX-SS-YYYY-MM-DDmat.zip

The XX-SS-YYYY-MM-DDdat.zip file should contain the dat-files, log-files, csv-files. The XX-SS-YYYY-MM-DDsac.zip file should contain the sac-files, net-files, fix-files. The XX-SS-YYYY-MM-DDmat.zip file should contain the mat-files. Be very conscientious about keeping these files, because they may save you a lot of unnecessary, extra work. It is customary to also store the neural network calibration files NET-files in the SAC.ZIP file, so that it is possible to calibrate the raw data always with the same calibration weights. There is no need to store and save the HV files, since the original raw DAT-files and NET-files provide all the information needed to generate them.

Cleaning the directory is a MUST, because the disk space is limited. If you are not immediately proceeding with the data analysis on this directory, the following files may be deleted: *.DAT, *.LOG, *.CSV, *.SAC, *.HV, *.MAT, *.FIX, *.NET.
What remains in the directory are the three ZIP files. Again, NEVER delete these ZIP files!
Especially the XX-SS-YYYY-MM-DDdat.zip file may be regarded as your life insurance!
If you want to continue with the data analysis, you typically only need the MAT-files. If you are also interested in the saccade traces, you will also need the *.DAT, *.SAC, and *.NET-files, or, alternatively, the *.HV and *.SAC-files.

How to clean using MATLAB

In Matlab you can use the function:

```
>> cleandir
```

This will create the three zip-files. It will also ask you whether you want to delete all other, non-MAT files (which it will do after you press ‘y’). Check whether this was succesul! Cleandir may not always succeed (and screw up your data) due to compatibility-issues.

7 Data analysis

Data analysis is performed with the help of the powerful program package MATLAB. In Appendix C an overview is given of the MATLAB functions that may be used for the data analysis. Often, however, you will to write your own analysis routines. In what follows, some general basic principles are given that are most commonly used in the analysis of eye and head movements.

7.1 Saccade and Stimulus Matrix

In the analysis of measured responses, the saccade- and the stimulus matrices, both stored in the MAT-files (see the previous chapter) are often needed. As soon as the MAT files have been generated with `>> hvtomat` or unzipped from an existing ZIP file, the MAT files can be loaded into MATLAB's workspace. For example, if you need the data from XX-SS-YYYY-MM-DD-0001.mat, you type:

```
>> load XX-SS-YYYY-MM-DD-0001.mat
```

The result is that two matrices will be stored in the workspace. You may verify this by typing the MATLAB command:

```
>> whos
```

Both the Stim and the Sac matrices will be listed:

```
>> whos
  Name            Size              Bytes  Class     Attributes

  Sac           323x20              51680  double
  Stim          648x11              57024  double
```

Sac Matrix

Convention
In each *row* of the Sac matrix *ALL* saccade parameters of *ONE* particular saccade are stored. Each *column* of the Sac matrix represents *ONE* particular parameter for *ALL* saccades.

In Appendix B the saccade parameters of each column are all listed. The table in Appendix B can also be obtained in MATLAB by typing:

```
>> index
```

Below, an example is given of a Sac matrix, for which only the first five columns are listed:

Row	Trial Number	Saccade number	Onset	Offset	Latency	…
1	1	1	310	340	220	…
2	1	2	460	470	520	…
3	2	1	320	360	240	…
4	3	0	220	244	40	…
5	3	1	380	400	360	…
6	4	1	294	340	188	…
7	4	2	450	466	500	…

8	6	1	300	342	200	…
Column	1	2	3	4	5	…

Thus, saccades from the same trial, but also from consecutive trials are listed in subsequent rows. The order is the same as was found by the detection algorithm (see ultradet). From the first and second columns it is seen that trials 1, 3 and 4 had more than one detected saccade, each with their own number, whereas in trials 2 and 6 only one saccade was detected. Trial 5 is omitted from the list, because there was no saccade detected in that trial.
Note that the first saccade of the third trial has received ranking number zero. The reason for this is that its latency was below 80 ms, in which case it can not be regarded as a stimulus-evoked saccade.

Stim Matrix

Convention
In each *row* of the Stim matrix *ALL* stimulus parameters of *ONE* particular trial are stored. Each *column* of the Stim matrix represents *ONE* particular parameter for *ALL* trials.

In Appendix B the stimulus parameters of each column are all listed. The table in Appendix B can also be obtained in MATLAB by typing:

```
>> index
```

Below, an example is given of a Stim matrix, for which only a part of the columns is listed:

Row	Trial Number	Target Number	Mode	Azimuth	Elevation	Eccentricity	…
1	1	2	0	0	0	0	…
2	1	3	1	20	30	37.2855	…
3	2	2	0	0	0	0	…
4	3	2	0	0	0	0	…
5	3	3	1	2	0	2	…
6	3	4	4	15	40	43.8631	…
7	4	2	0	0	0	0	
8	4	2	1	2	30	30.1809	
9	5	2	0	0	0	0	
10	5	2	1	40	10	41.7460	
11	6	2	0	0	0	0	
12	6	2	1	-20	5	20.6679	
Column	1	2	3	4	5	…	…

Again, subsequent trials are listed below one another, however parameters of subsequent stimuli within each particular trial are also listed below one another.
The first column tells us that there have been three trials, whereas the second column indicates that there have been a various number of stimuli per trial. In trial 1, the first stimulus (Target Number 2, as nr 1 is reserved for data acquistion start), a visual fixation point at the center of FART (Mode = 0,column 3), Azimuth and elevation both 0, (columns 4 and 5). The second stimulus was a peripheral target (columns 4 and 5) and was an auditory target (Mode = 1). Trial 3 had 3 stimuli (Column 1), of which the First was visual (Mode = 0), the second was auditory (Mode = 1) and the third was a skyLED (Mode = 4).
Note that, as a rule, the Sac and Stim matrices do not have an equal number of rows, since each trial

may contain any number of detected saccades. In the analysis it is therefore important to keep this in mind, and always find a way to associate a given saccade with the correct trial number and stimulus events. Below, some hints are given on how to achieve this.
Sometimes in an experimental session the same stimulus configurations have been presented, but these are listed in different files. With the MATLAB function:

```
>> loadmat(MatFiles)
```

it is possible to load the relevant MAT files with one keystroke. For example, if you want to combine the MAT files from `XX-SS-YYYY-MM-DD-0001.mat`, `XX-SS-YYYY-MM-DD-0002.mat` and `XX-SS-YYYY-MM-DD-0003.mat`, you type:

```
>>[Sac,Stim]= loadmat(['XX-SS-YYYY-MM-DD-0001';' XX-SS-YYYY-MM-DD-0002';' XX-SS-YYYY-MM-DD-0003']);
```

The result will be one big Sac and one big Stim matrix in which the data are put into the subsequent rows.

7.2 Saccade Selection

After loading the appropriate Sac and Stim matrices, the first step is typically some kind of selection procedure. For example, in your analysis you may be interested in the primary saccade responses toward the stimulus in each trial only, and to disregard the correction saccades. In MATLAB such a procedure is readily implemented through the use of so-called selection vector, or with index vectors. The definition of the selection criteria is most often implemented as a logical statement, involving logical operators like & (AND), || (OR), == (EQUAL), > (LARGER), < (SMALLER), etc.

Selection vectors
Suppose you want to select all primary saccades. An appropriate selection vector for primary saccades, `>>sp`, is then created by looking at the saccade number in the Sac matrix (column 2). Type:

```
>>sp  = (Sac(:,2)==1);
```

The selection vector is a column vector, containing the SAME number of rows as the Sac matrix, in which the values are only ZERO (the logical operation was FALSE) and ONES (when TRUE).
For our previous example this would yield:

Selection	Trial Number	Saccade number	Onset	Offset	Latency	…
1	1	1	310	340	220	…
0	1	2	460	470	520	…
1	2	1	320	360	240	…
0	3	0	220	244	40	…
1	3	1	380	400	360	…
1	4	1	294	340	188	…
0	4	2	450	466	500	…
1	6	1	300	342	200	…
Column	1	2	3	4	5	…

Now suppose that you are interested in the LATENCY of the primary saccades (column 5). By using the selection vector, `>>sp`, it is straightforward to generate a `>>Lat` vector with only the

latencies of the primary saccades, as follows:

```
>> Lat = Sac(sp==1,5);
```

In plain language, this command says: “Return all values from the fifth column of the >>Sac Matrix (containing the latencies) from those rows for which the corresponding row in the sp vector equals 1”.
The values of the >>Lat vector are therefore:

```
>> Lat=[220, 240, 360, 188, 200]
```

Index vectors

An alternative way to achieve the same result as in the previous example is by using an index vector. The MATLAB command >>find generates such a vector. In our case:

```
>>ip = find(Sac(:,2)==1);
```

In plain language: “provide the row_indices for the second column in matrix >>Sac whenever the value is equal to one”. The result of this operation yields the rows in which the primary saccades are listed, thus

```
>>ip = [1,3,5,8]
```

Subsequently type:

```
>>Lat = Sac(ip,5);
```

yielding exactly the same >>Lat vector. Both methods yield entirely equivalent results (although MATLAB is slightly faster with selection vectors).
Typically, saccades have a reasonably short latency, around 200 ms. When the latencies are much longer than that, this is an indication that the subject is not very alert, and therefore may respond in an otherwise sloppy way. For this reason it is often desirable to add an extra criterion to selected saccades, namely to exclude saccade responses with latencies exceeding, say, 350 ms.
To create a selection vector or, alternatively, an index vector for this criterion type:

```
>> sl = (Sac(:,5)<=350);
>> il = find(Sac(:,5)<=350);
```

The latencies of the saccades in our example that fulfill the 350 ms criterion are generated by one of the following commands:

```
>>Lat  = Sac(sl==1,5);
>>Lat  = Sac(il,5);
```

The result is (check for yourself): >>Lat=[220,240,40,188,200]
Note, that the saccade with the 40 ms latency is now also included. This is the saccade with rank number 0 in trial 3.
If you want to select

a) only primary saccades AND
b) latencies less than 350 ms

both selection criteria should be applied together. By far the simplest way to achieve this is to combine the selection vectors >>sp and >>sl into a new selection vector,>>sel:

```
>>sel = sp & sl
```

The resulting vector now contains a 1 only when both the sp vector AND the sl vector contain a 1 in the same row. The final selection that meets our goals is therefore

```
>>Lat = Sac(sel==1,5);
```

with the result: >>Lat=[220,240,188,200]

Note that it is relatively simple to combine selection vectors in order to impose several simultaneous constraints onto your data. The index vector is not a very useful tool for combining different criteria.

7.3 Selection of Stimulus Parameters

This section explains how to combine selected saccades with the corresponding stimulus events that evoked them. As an example, suppose you want to investigate the relation between the target eccentricity and primary-saccade latency. First, select the primary saccades that also fulfill the latency criterion (Lat ∈[80,350] ms: see above). Then, we need to know the eccentricities of the associated targets. To that means, we first determine the trial numbers to which the selected saccades belong (obtained with selection vector, `sel`), by generating a trial-number vector, `Tnr`:

```
>>Tnr = Sac(sel,1);
```

In our example, we find: `>>Tnr = [1,2,4,6]`. This tells us from which trials the selected saccades originate. To extract the desired stimulus parameters from the Stim matrix, we need to find these trial numbers in the first column. We can do this by using the Matlab function `ismember`:

```
>> selTnr = ismember(Stim(:,1),Tnr);
```

selTnr can be used as a selection vector for the Stim matrix. Therefore, type:

```
>>Exc = Stim(selTnr,6);
```

to obtain a vector Exc with the eccentricities of all targets (which are to be found in column nr. 6). To obtain a vector with the eccentricties of only the peripheral acoustic targets, we also need to select for stimulus type:

```
>> selType = Stim(:,3) == 1;
```

And combine this with the previous selection vector:

```
>> selexc = selType & selTnr;
>> Exc = Stim(selexc,6);
```

In the example, `>>Exc = [37.2855, 30.1809, 20.6679]`. The latencies of the associated saccades were obtained with the command:

```
>>Lat = Sac(sel==1,5);
```

resulting in `>>Lat=[220,240,188,200]`.

7.4 Loading Signals

Often it is desired to also analyse the entire movement trace, instead of only certain parameters. Usually we will be only interested in the actual saccades (plus and minus a restricted time window, e.g. from 100 msec before saccade onset until 200 msec after offset, etc). This feature is possible with the MATLAB function

```
>>fgetsac(DatFile,Nchan,Chan,SacFile,Type,Pre,Post,Dur)
```

By using the SAC file, this function selects the pieces of data around the detected saccades. Saccade traces may be loaded from either the raw DAT files, or from the calibrated HV files. For example, to load the saccades from DAT file `XX-SS-YYYY-MM-DD-0001.dat` with SAC file `XX-SS-YYYY-MM-`

DD-0001.sac, type:
```
>>[Sh,Sv]=fgetsac('XX-SS-YYYY-MM-DD-0001.dat',2,[1,2],' XX-SS-YYYY-MM-DD-0001.sac','sac',0,0)
```

Each row of the two matrices Sh and Sv contains the horizontal and vertical position signals of each saccade, respectively. Note, that as long as the Sac matrix has not been changed, each row of this matrix corresponds to the same row in both Sh and Sv. By sticking to this procedure, it is now straightforward to select saccade traces by using the same selection vector sel as has been used for the Sac matrix.

Instead of selected time windows around saccades, it is also possible to load the entire trial. This is done with the function `>>fgettrl(DatFile,Nchan,ChanNr)`. In our example, type:
```
>>[Sh,Sv]= fgettrl(('XX-SS-YYYY-MM-DD-0001.dat',2,[1,2]);
```

Note that this function assumes a default trial length of either 750 samples 1000 samples. If you have a different trial length (e.g. 2250 samples long), you can use:
```
>>[Sh,Sv,Sf]= fgettrl('XX-SS-YYYY-MM-DD-0001.dat',2,[1,2,3],2250);
```

When data are loaded from HV files, calibration is not needed. However, if the data are extracted from the DAT files (which is usually the case, since HV files are not stored), the signals should first be calibrated. Calibration is done by evoking the MATLAB function `>>calib(h,v,calfile)`. This functions uses a neural network whose parameters are stored in the NET file:
```
>>[Sh,Sv] =  calib(Sh,Sv,Sf,'XX-SS-YYYY-MM-DD-0000.net');
```

Yet another way of loading the entire data file is by using the command:
```
>>DataVector = loadseq('XX-SS-YYYY-MM-DD-0000.hv','float32',inf,'l')
```
or
```
>>DataVector = loadseq('XX-SS-YYYY-MM-DD-0000.dat','float32',inf,'l')
```

7.5 Go Ahead

At the start of an analysis session, all the data are first loaded from the MAT files. Then, if needed saccade traces are loaded (and calibrated). As soon as this is done, it is customary to first select all primary responses. Since the analysis is typically based on first-saccade responses, the remaining data are then deleted.
This part of the analysis can be done by evoking the `>>firstsac` function. This procedure makes subsequent calls to four sub-routines: `>>sortsac`, `>>linksac`, `>>primsac`, and `>>checklat`.

Procedure `>>sortsac` sorts saccades according to onset latency.
Procedure `>>linksac` ensures that the head-saccade ranking numbers correspond with the right gaze and eye saccades (see above).
Procedure `>>primsac` selects all primary saccades and deletes the remaining responses.
Procedure `>>checklat` performs a check on the latencies. Saccades with latencies that are either too short or too long are also discarded.

If you are also interested in the actual saccade traces, it is advisable to load all saccades before the `>>firstsac` procedure is evoked. The advantage is that the same selection procedures can be performed on the Eh, Ev, Hh, Hv, Gh and Gv matrices.

Matching the Stim and Sac matrices:

```
>> Stim = matchsac(Sac,Stim);
```

If you are interested in a single stimulus per trial, you could combine the Sac and Stim matrices with:

```
>>SupSac = supersac(Sac,Stim);
```

This SupSac matrix will contain per trial all saccade parameters in the first 20 columns, similar to the Sac matrix. The last 10 columns will contain the stimulus parameters as in the Stim matrix, except for the trial number. By default >>supersac will only add the stimulus parameters of the first sound stimulus in every trial. You can change this by supplying some additional input:

```
>>SupSac = supersac(Sac,Stim,0,2);
```

The first additional input argument is the modality of the stimulus (0 = LED, 1 = SND) and the second additional input argument contains the number of the stimulus type. Thus, the above command will look for the 2nd LED in every trial.

8 Sound measurements

8.1 Bruel & Kjaer Sound Level Measurement

What to do – max input and min output section gain, A-filter, 22.4 Hz filter HP-filter, fast averaging, RMS, AC output, Preamp Input

8.2 HRTF Set-up

Luxman (max gain- 0dB when measuring in ear canals, CD in, CD straight) – Krohnhite (LP 20 kHz, 20 dB Gain) – RP2-1, RP2-2 inputs, [Left Microphone:left channel:channel 1: upper channel] or [Right Microphone:right channel:channel 2:lower channel]

Placement of microphones

8.3 The HumanV1 Program to measure HRTFs

8.4 Generating stimuli and log-files

Snd999.wav – Schroeder Sweep
Snd998.wav – Golay Code

8.5 HRTF Analysis

Schroeder sweep
Golay code
Impulse/gwn

Gensweep
Getspectrum
Plotdtf

9 Spike Data Analysis

10 FAQ/Known Issues

The sampling rate does not change when I alter the CFG file

This is because it is presently fixed.

The number of samples

Appendix A: File Extensions

Appendix B: Matrix Structures

Appendix C: Matlab Functions

C.1 Files & Directories

datadir	- Get current data directory name (fileparts)
fcheckext	- Check file extension
fcheckexist	- Check for existence of filename
gotodat	- goto specified datadirectory (you have to change gotodat so that it references your own data-directory)
hrtf	- goto HRTF-data-directory (in [gotodat]HRTF)

C.2 Calibration

readlog

calrose

gencal

C.3 Saccade Analysis

hvtomat	- Extract saccade parameters from hv- and sac-files, and save in mat-files
hvfilt	- Low-pass filter the eye/head/gaze-traces in hv-files
ultradet	- Interactively detect saccade on- and offset

getvel

C.4 Coordinate Systems

Built-in:
Cart2pol
Pol2cart
Cart2sph
Sph2cart

Tom:
Deg2rad
Rad2deg

Marc
Azel2cart
Azel2pol
Fart2azel

Azel2fart

Paul& Jeroen:
Azelrphi
Rphiazel
Xyzrphi
Rphixyz

Sellat
Split
Firstsac

C.5 Stimulus Generation

Gengwn

C.6 Statistics

Rnduni
rndval