HUTear Matlab Toolbox *version 2.0*

Aki Härmä

March 7, 2000

HUTear is a MATLAB toolbox for auditory modeling. This manual is also available as a single PostScript file at http://www.acoustics.hut.fi/software/HUTear/HUTear.ps.gz HUTear.ps.gz. This version of the toolbox has been published in Matlab DSP conference and the article is available here: http://www.acoustics.hut.fi/software/HUTear/MatlabDSP.ps http://www.acoustics.hut.fi/software/HUTear/MatlabDSP.ps.

1 Introduction

The components of the HUTear toolbox were originally designed for the experiments reported in an internal report http://www.acoustics.hut.fi/~aqi/papers/Time.ps.gz, *Temporal masking effects: single incidents* at Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technologyhttp://www.acoustics.hut.fi However, since there were some interest among the researchers of the laboratory for a these tools we decided to collect them into a toolbox and make it available for anyone at http://www.acoustics.hut.fi/software/HUTear http://www.acoustics.hut.fi/software/HUTear.

There exists already at least two more carefully designed and more versatile auditory modeling packages, e.g, Auditory Image Model http://www.mrc-cbu.cam.ac.uk/personal/roy.patterson/aim/ and Development System for Auditory Modelling (also known as *lutear*) http://www.essex.ac.uk/psychology/hearinglab/lutear/home.html. However, it is quite difficult to use those packages with MATLAB, even if a MATLAB front-end is available at least for the former. We have also encountered some problems in compiling those packages on different platforms. Malcolm Slaney's new Auditory Toolbox version 2 http://www.interval.com/ malcolm/pubs.html is also available and it contains several blocks which are not in the current version of HUTear.

HUTear package uses some functions from a MATLAB toolkit by Toshio Irino http://www.hip.atr.co.jp/~irino/GCFB/GCFBpack.html. Some of his functions have been included into the HUTear package. If you want to use (optional) time-varying *gammachirp* filterbank with the model you must download his package or use our implementation. B. J. C. Moore from the University of Cambridge, Auditory Perception Grouphttp://hearing.psychol.cam.ac.uk/ have also kindly provided code fragments which are used in a loudness model function [1].

2 Installation

Basically, you should download all, compile MEX files, and install them so that MATLAB can find them.

2.1 UNIX platforms

RedBall

Download HUTear2.tar.gz http://www.acoustics.hut.fi/software/HUTear/HUTear2.tar.gz to the place where you want to install it.

Gunzip and untar the package:

```
>> gunzip HUTear.tar.gz
>> tar xf HUTear.tar
```

Edit makefile if necessary. It is possible that you need to change **MEXEXT=mexsg** line to you are not using a SGI's workstation.

Compile MEX files

» **make all**

Set path in MATLAB so that the functions (**HUTear**-directory) can be found. In addition set path to **examples**-directory in order to make ready-made profile description files to work.

Go to **examples** directory.

Run **HUTear_intro** in MATLAB to see that everything works correctly.

You may run also demonstration scripts **demo1**, **demo2**, etc.

2.2 Other platforms

GreenBall

Download HUTear2.tar.gz http://www.acoustics.hut.fi/software/HUTear/HUTear2.tar.gz or HUTear.zip http://www.acoustics.hut.fi/software/HUTear/HUTear.zip to the place where you want to install it.

Decompress and extract the package using tools you have.

Compile the MEX files from **source** directory (consult your Matlab manual how to do that).

Place the compiled MEX binaries in the **HUTear** directory.

Set path in MATLAB so that the functions can be found.

Go to **examples** directory.

Run **HUTear_intro** in MATLAB to see that everything works correctly.

You may run also demonstration scripts **demo1**, **demo2**, etc.

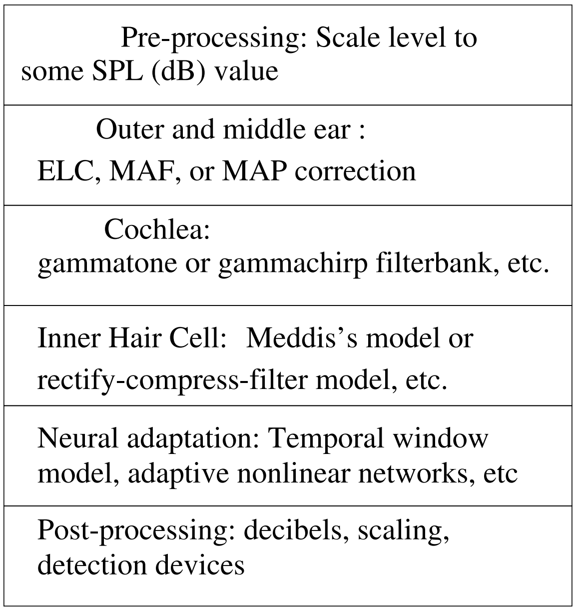

Figure 1: A block diagram of a generic auditory model

3 Structure of the toolbox

The HUTear toolkit is a modular package which contains a generic frame for an auditory model and a set of modules which can be hooked into it to produce a model for some particular application. The basic building blocks of such a model are shown in Fig. ??.

The name of the frame function is **AudMod**, which is used in the following way:

» **A=AudMod(signal,profile);**

where `signal` is a single row or column vector of signal samples. `profile` is is either a struct variable or a name of a text file where a struct variable is defined. The output `A` is a matrix of output row vectors. The number of output channels is specified in profile description. It is the same as the number of channels in the filterbank. Using only one, e.g., gammatone filter, the model is reduced to a single channel model. A user may design a new model profile or use some of the *ready-made profiles*, (see below.)

4 Profile description examples

As mentioned above, the particular auditory model which is implemented using **AudMod** may be defined in a profile description file. A typical profile file is shown below:

```
% Profile_Plack98
%
% This profile corresponds to a revised temporal window model
% presented in (Plack&Oxenham, 1998).
%
% This file is a part of HUTear- Matlab toolbox for auditory
% modeling. The toolbox is available at
% http://www.acoustics.hut.fi/software/HUTear/

% Copyrights: Aki Härmä, Helsinki University of Technology,
% Laboratory of Acoustics and Audio Signal Processing,
% Espoo, Finland.
% Date: August 20 1999
% Email: Aki.Harma@hut.fi
clear model;

% General parameters
model.id = 'Temporal window model (Plack & Oxenham, 1998)';
model.fs = 44100; % Sampling frequency
model.ds = 1;     % Outputs downsampled to rate fs/ds

% Outer and middle ear filtering
model.outmid.id='Fixed filter';
model.outmid.file='MAF255';

% Model for inner ear
model.cochlea.id='Gammatone filterbank';
model.cochlea.gt.design='gammatone';
model.cochlea.gt.nch=64;
model.cochlea.gt.frange=[200 22050];

% Mechanical to neural - haircell
model.haircell.id = 'Half-wave rectification, compression, and _filtering';
% Rectification
model.haircell.rcf.r='half';
% Compression
model.haircell.rcf.c='Plack98';
% Smoothing
model.haircell.rcf.f='no';
% Thresholding
model.haircell.rcf.thrhld='no';

% Neural adaptation
model.neural.id ='Sliding temporal window function';
model.neural.function ='Plack98';
```

Profile_file	Modules	Reference
Profile_Plack98.mProfile_Plack98.m	Gammatone filterbank Half-wave rectification Nonlinear compression Sliding temporal integrator	[2]
Profile_Dau96.mProfile_Dau96.m	Gammatone filterbank Half-wave rectification Low-pass filtering Püschel's adaptation network	[3]
Profile_Karjalainen96.mProfile_Karjalainen96.m	Gammatone filterbank Half-wave rectification Low-pass filtering Karjalainen's adaptation network	[4]
Profile_Meddis91.mProfile_Meddis91.m	Gammatone filterbank Meddis's IHC model	[5]

Table 1: Ready-made profile description files

```
% Post-processing
model.pp.id = 'Decibel scale';
model.pp.mapping='decibel';
```

It simply sets up a struct variable which control the the **AudMod** function. This profile description file defines a *temporal window model* introduced in [2]. The fields have been listed in 5.

One may use this with `AudMod` in the following two ways:

1. Load model struct into Matlab's workspace and use it with `AudMod`:

   ```
   >> Profile_Plack98;
   >> A=AudMod(signal,model);
   ```

2. Call `AudMod` so that the profile description file is used as an argument. In this case, `AudMod` loads the struct parameter and it is not visible in Matlab's workspace.

   ```
   >> A=AudMod(signal,'Profile_Plack98');
   ```

There are four ready-made profile description files in the HUTear package. One may use them directly, or modify to produce a new model. The files are listed in Table 1 and they are placed in **examples** directory.

5 Structure of the profile description

The struct **model** has five main fields where the elements of an auditory model are defined. Any of these fields may be omitted. If **model** is an empty struct,

model.outmid	Outer and middle ear model, e.g., a fixed filter
model.cochlea	Cochlear filtering, e.g., a gammatone filterbank
model.haircell	mechanical-to-neural transduction, e.g., an inner hair cell model
model.neural	Neural processing, e.g., model for neural adaptation
model.pp	Post-processing, e.g., scaling or mapping to a decibel scale

an input signal passes **AudMod** unchanged. Currently, there is only one *global* parameter, **model.fs**, which defines the sampling rate used in all computational elements of the system. The parameter **model.ds**, which appears in the **Profile_Plack98** above, is not functional in the current version of HUTear. All ready-made profile files also have several text fields, e.g., **model.id** and **model.cochlea.id** which have no meaning in **AudMod** but are informational when working with the struct in Matlab's workspace.

The five main fields are

All main fields have several sub-fields.

5.1 Structure of model.outmid

This field is responsible for modeling of the outer and middle ear. Currently, two alternative sub-fields can be defined.

1. If this field is defined as

   model.outmid.file='filename';

   it is assumed that this stage is modeled using a fixed linear filter. **AudMod** loads in the file named **filename.mat** and uses the obtained *filter coefficients* to filter the signal using the following Matlab command `signal=filter(a,b,signal);`. It is important that the names of the forward and backward filter coefficients are `a` and `b`. Typically, the filter coefficients can be designed using **Make_OEM** function which is introduced later in this document.

   Using **Make_OEM** it is also possible to design *frequency-warped* outer and middle ear filters. To utilize this feature, one have to download **WarpTB** http://www.acoustics.hut.fi/software/warp/WarpTB/ package which is a set of routines for frequency-warped signal processing.

2. Alternatively (you cannot use both),

   model.outmid.function='filename';

   means that outer and middle ear filtering is replaced by another Matlab function (filename.m) which has the following form:

   signal=MyFunction(signal);

5.2 Structure of model.cochlea

In hearing, frequency analysis is performed in *cochlea* which is placed in the inner ear. There are several approached for modeling the performance of this organ. In the current version of this package, cochlea is typically modeled using so called gammatone filterbank. There are three *alternative* sub-field systems available and one additional (optional) field. The number of frequency channels at this phase typically determines also the number of channels at the output of **AudMod**.

1. If there is a field

   model.cochlea.gt

   AudMod uses **Gammatone** function which contains the design and implementation of a gammatone filterbank. The parameters of this filterbank are transmitted in the sub-fields of **model.cochlea.gt**. This is used in all ready-made profiles. However, it may be computationally inefficient because it also contains the design of the filterbank. The design parameters for **Gammatone** function are introduced later in this document.

2. In using a gammatone filterbank model for cochlear filtering it is suggested to design a filterbank using **Make_GTbank** routine (see below) in advance, store the filter coefficients into a .mat-file, and then define the cochlea field in the following way:

   model.cochlea.fb='filename'

   where **filename** is a .mat file containing the coefficients of the filters as row vectors in matrices **f** and **b**.

3. Alternatively,

   model.cochlea.function='filename';

   means that the cochlear processing is replaced by another Matlab function (filename.m) which has the following form:

   signal=MyFunction(signal);

   This might implement, e.g., a transmission line model of the cochlea.

4. It is possible to add a time-varying asymmetric compensation filterbank to the output channels a cochlear model. This is based a model introduced by Irino and Patterson [6]. The function is activated if the field

   model.cochlea.asymmcomp

   exists.

5.3 Structure of model.haircell

There are three different approaches to implement the mechanical-to-neural transduction which is the function of the inner haircells on basilar membrane.

1. In many auditory models, this phase is implemented as a *rectify-compress-filter* system. This function is activated in **AudMod** if the following field exists:

   model.haircell.rcf

   There are three sub-fields defined:

   (a) **model.haircell.rcf.r** defines the type of rectification. There are three alternatives: 'full', 'half', and 'no', implementing full-wave rectification, half-wave rectification, or no rectification, respectively.

   (b) **model.haircell.rcf.c** defines the compression function that is applied to the rectified channel signals. Currently, there are two alternatives: **model.haircell.rcf.c='Plack98'** which implements a nonlinear compression function introduced in [2] and **model.haircell.rcf.c=c** which implements a classical compression rule $y = x^c$ for all sample values. Typically, $c = 0.7$.

   (c) **model.haircell.rcf.f** has currently two alternatives. If **model.haircell.rcf.f='1kHz'** the channel signals are filtered using a lowpass filter with a cut-off frequency at 1 kHz (for a sampling rate given in **model.fs**). Alternatively, **model.haircell.rcf.f='tw20'** causes the rectified and compressed channel signals to be filtered using a first order integrator with a time constant of 20 ms.

2. If field **model.haircell.ihc** exists, **AudMod** uses Meddis's inner haircell and synapse model, see, e.g., [5]. By default, Meddis's parameter set for *medium spontaneous rate fibres*. It is also possible to define other parameter sets in an additional field **model.haircell.ihc.params**. This is given as a parameter vector and its structure becomes clear by comparing [7] with the source file Meddis_network91M.c. In future versions of this package there are going to be automatic functions for the design of different parameter sets for Meddis's model. In addition, if **model.haircell.ihc.refractory** exists, the outputs of the IHC model are processed using a model for neural refractoriness [5].

3. As in other main fields it is possible to use

   model.haircell.function='filename'

   to specify an alternative function which is used to model mechanical-to-neural transduction.

4. In some cases it is convenient to threshold the channel signal at this phase, e.g., to get rid of negative or zero values. If **model.haircell.threshold=b**, all signal values *below* **b** are changed to **b**.

5.4 Structure of model.neural

There is currently implementation for four different *neural adaptation models* in the package. There is only one sub-field which may have the following values:

1. **model.neural.function='Meddis91'; AudMod** uses Meddis's model [5] for neural refractoriness. This is the same element which also appears if **model.haircell.refractory** exists.

2. **model.neural.function='Plack98'** This is an efficient implementation of a *temporal window model* of neural adaptation introruced in [2].

3. **model.neural.function='Dau96'** This uses a nonlinear adaptation network introduced in [3]. If this model is chosen, one must also set **model.neural.Dau.thr** to some small positive value.

4. **model.neural.function='Karjalainen96'** This implements a nonlinear adaptation network which was introduced in [4].

5. As in all other main fields

   model.neural.function='filename'

   can be used to define a new model for this stage.

5.5 Structure of model.pp

Currently, there are two alternatives.

1. **model.pp.mapping='decibel'** causes the output values of the previous stages to be mapped to the decibel scale.

2. **model.pp.mapping='filename'** is essentially the same as the **function** sub-field in other main fields.

6 Additional functions

6.1 Scaling of the input signal

Most of the nonlinear elements of an auditory model, e.g., at the level of mechanical-to-neural transduction and neural adaptation are level-dependent. Therefore, it is necessary to fix the amplitude range and its relation to the sound pressure level of a real acoustic signal. Here, we adopt use a convention that the *root-mean-square*, RMS, value 1 in the digital signal corresponds to the sound pressure level of 0 dB. The function **Pascalize** can be used to scale a steady-state signal such that it corresponds to a sound pressure level of A dB. For example,

» **ssig=Pascalize(sig,60);**

scales a signal *sig* such that its sound pressure level is 60 dB.

6.2 Design of gammatone filters

If the gammatone filterbank is defined in the field **model.cochlea.gt**, see above, the function **Gammatone** designs a set of filter and use them to do the cochlear filtering. It is usually more efficient to design the filterbank in advance, store the filter coefficient in a file and use **model.cochlea.fb.file** field to specify the filterbank.

Make_GTbank is an interactive function which can be used to design various types of gammatone filterbanks. This is called by

» **[f,b]=Make_GTbank;**

The function asks first user to specify the type of the filterbank. There are three alternatives: (1) conventional gammatone implementation is the one introduced by Malcolm Slaney [8]. The second and third alternatives are a (2) fourth-order and a (3) sixth-order complex-valued gammatone filterbank for which HUTear has an efficient implementation which uses **cgtbank.mex**** function. Derivation of these filters can be found HereGammatone/Complex$_{g}t.html$.

Next, the user is asked to give sampling rate in Hertz. After that, the function gives three alternatives: (1) specify all center frequencies in Hertz, (2) specify all center frequencies in ERB (1-42), and (3) give the number of channels and the lowest center frequency.

After that, the user is asked if he/she wants to store the coefficients directly to a file or to just take the coefficient matrices to Matlab's workspace. **AudMod** assumes that the coefficients are row vectors in matrices which named **f** and **b** as above.

6.3 Design of an outer and middle ear filter

As shown above, the model for outer and middle ear is usually a linear filter. The coefficients of the filter are in a mat-file which is set by the **model.outmid.file** field. This filter may be designed using the **Make_OEM** function.

For example,

```
>> [a,b]=Make_OEM(44100,255,'ELC','FIR');
>> save filename.mat a b
```

designs a 255th order FIR approximation at 44100 Hz sampling rate which is matched with the Equal Loudness Curves at the loudness level of 0 phon. The coefficients are stored here in a mat-file (filename.mat). It is important that the coefficient vectors are named **a** and **b**. Otherwise, **AudMod** is not able to identify the filter coefficients.

It is possible to design also IIR filters using the same function, see **help Make_OEM**. In addition, if WarpTBhttp://www.acoustics.hut.fi/software/warp is available, it is also possible to design and use frequency-warped FIR and IIR filters.

There is an interactive graphical user interface for **Make_OEM** in the toolbox. It is called by

» **OEMtool**

It can be used to compare different implementation technique as a function of model order and finally to store the coefficients to a named file.

Acknowledgements

This work has been supported by the *Academy of Finland* and *Jenny ja Antti Wihurin rahasto*. The authors are also grateful to Brian C. Moore, Matti Karjalainen, Toshio Irino, and Malcolm Slaney for permission to use some of their code fragments and functions in the toolbox.

References

[1] B. C. J. Moore, *Introduction to the psychology of hearing.* Academic Press, 4th edition ed., 1997.

[2] C. J. Plack and A. J. Oxenham, "Basilar-membrane nonlinearity and the growth of forward masking," *J. Acoust. Soc. Am.*, vol. 103, pp. 1598–1608, March 1998.

[3] T. Dau and D. Püschel, "A quantitative model of the effective signal processing in the auditory system. I. model structure," *J. Acoust. Soc. Am.*, vol. 99, pp. 3615–3622, June 1996.

[4] M. Karjalainen, "A binaural auditory model for sound quality measurements and spatial hearing studies," in *Pro. ICASSP'96*, vol. 2, (Atlanta), pp. 985–988, IEEE, May 1996.

[5] R. Meddis and M. J. Hewitt, "Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I:pitch identification," *J. Acoust. Soc. Am.*, vol. 89, pp. 2866–2881, June 1991.

[6] T. Irino, "A time-varying, analysis/synthesis auditory filterbank using the gammachirp," in *Proc. ICASSP'98*, (Seattle), pp. 3653–3656, IEEE, 1998.

[7] R. Meddis, M. J. Hewitt, and T. M. Shackleton, "Implementation details of a computational model of the inner hair-cell/auditory-nerve synapse," *J. Acoust. Soc. Am.*, vol. 87, pp. 1813–1816, April 1990.

[8] M. Slaney, "An efficient implementation of the Patterson-Holdsworth filter bank," Apple Technical Report 35, Apple, 1993.