SPro3.2
User Manual

Guillaume Gravier

Last update, Dec. 99

1 Introduction

This user manual briefly describes the main features of the speech processing toolkit SPro3.2. Basically, SPro was designed for variable spectral resolution analysis of the speech signal (see section 1.1 for more details on this topic). It provides a library for the standard (frame-level) analysis techniques and for the I/O functions. Runtimes for feature extraction of a speech signal were created using the library.

In section 2, the file formats used by SPro are described. In section 3, the basic feature extraction tools are described. Finally, an overview of the functions provided by the library is given in section 4.

1.1 Variable resolution spectral analysis

This technique is fully described in [1] where it is applied to speech for coding purposes. Attempts to use this technique in speech and speaker recognition are described in [2, 3].

Briefly, it consists in applying a bi-linear transformation of the frequency scale defined by

$$\overline{\omega} = \arctan \left| \frac{(1-\alpha^2)\sin\omega}{(1+\alpha^2)\cos\omega - 2\alpha} \right|$$

Spectral transformations for various values of α are plotted in figure 1 where the x-axis is the original frequency scale (ω) and the y-axis, the transformed frequency scale ($\overline{\omega}$).

2 File formats

2.1 Signal

SPro supports two signal file formats. The default format is the *linear 16* format, that is, raw PCM data with 16 bits per samples. In each runtime, the default sample rate of 8 KHz can be changed. If required, SPro can be linked with the SPHERE 2.6a library, in order to directly read SPHERE signal files[1]. In this

[1]For this, SPro must be compiled with -DSPHERE. SPHERE 2.6a is freely distributed by the American National Institute of Standards and Technologies (NIST) and can be found at URL *http://www.itl.nist.gov/div894/894.01/software.htm*.

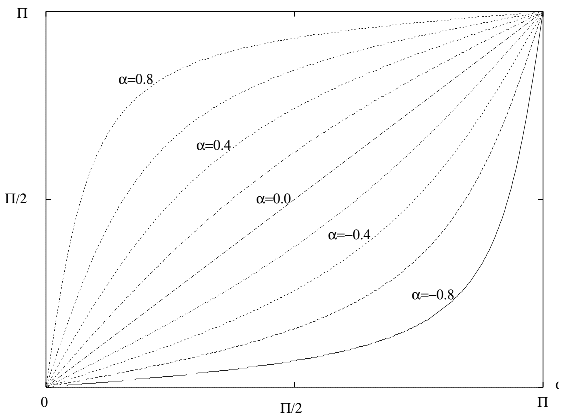

Figure 1: Bi-linear transformation of the frequency scale

case, the sample rate is directly read form the SPHERE header but all the other information of the header are lost.

2.2 Data

A data file is the file that contains the feature vectors. It consists of a 16 bytes header followed by the concatenated feature vector. The header is defined by the four following fields:

- data kind (4 bytes), where data kind is an enumeration of the following: OTHER, FBANK, FBCEPSTRA, LPCEPSTRA, LPCOEFF, PARCOR and LAR[2]
- basic feature vector dimension (4 bytes)
- number of feature vectors (4 bytes)
- qualifier (4 bytes)

The qualifier field is a bit field used to describe in more the details the content of the file. The bits have the following meanings:

1. log energy appended
2. long term mean of the data subtracted
3. absolute log energy suppressed (this bit makes no sense without bit 4)
4. Δ coefficients appended
5. $\Delta\Delta$ coefficients appended

It must be noted that the second header field contains the basic feature vector dimension (*i.e.* the dimension of the static coefficients regardless of the energy) and not the real feature vector dimension which must be retrieved according

[2]Note that the order of the enumeration have changed from version 3.0!

to the qualifier field. The feature vectors are stored as *float* (4 bytes) with the static coefficients first, optionally followed by the log-energy, the Δ and $\Delta\Delta$ coefficients.

3 User guide

In this section, the runtimes are briefly described. A short on-line description can always be obtained using the *-h* option. Every runtime also have a *-v* switch to run in verbose mode and a *-V* switch to print out the version number. The four runtimes which take a signal as input (*sfbank*, *sfbcep*, *slpc* and *slpcep*) shares some common options described below:

- *-f num*: set sample rate of the input signal (default is 8,000 Hz).
- *-S*: indicates the input signal file is a SPHERE file.
- *-w str*: set the analysis window type. *str* can be *none*, *haming*, *hanning* or *blackman* (default is *haming*).
- *-l num*: set the analysis window length (default is 20.0 ms).
- *-d num*: set the analysis window decay (default is 10.0 ms).
- *-a num*: set the spectral resolution factor α (see sec. 1.1).

Except for *slpc*, the following options are also common:

- *-k num*: set coefficient of the $1 - kz^{-1}$ pre-emphasis filter (default is 0.95).
- *-e*: output log-energy
- *-s*: scale log-energy according to $E = (E - E_{\max})\ s + 1$, where s is specified by the *-s* switch and $E_{\max}$ is the maximum energy in the file.

3.1 scopy

Usage: scopy [options] ifile ofile

Copy of a data file *ifile* into *ofile*. One can specified the start and/or end feature vector of the copy using the *-e num* and *-s num* switches, thus allowing the copy of a part of the input data file. The input data can be converted using the *-C str* option. The argument to this option is a string composed of one or zero occurrence of the letters D, A, N and Z. The meaning of the letters is:

- D: add Δ coefficients
- A: add $\Delta\Delta$ coefficients
- N: suppress log-energy. This is only valid if the log-energy is in the input and if D is specified or if the input file already contains the Δ coefficients.
- Z: long-term mean subtraction

The default is to set the output file format to the SPro format but this can be changed using the *-o str* option where *str* can be ASCII (useful to do Matlab or gnuplot graphics), htk1 or htk2 to save files in HTK v1.x or HTK v2.x format. In HTK v1.x, SPro finds out the most appropriated sample kind. In HTK v2.x, the sample kind is set to other. See [4] for more details on HTK file formats.

3.2 slist

Usage: slist [options] datafile

List the content of a data file on screen. The *-s num*, *-e num* and *-C str* options are the same as for *scopy*. The *-i* options, suppress the listing of the feature vectors thus allowing to see only the header information.

3.3 sfbank

Usage: sfbank [options] infile outfile

Filter-bank analysis of a speech signal. This program takes as input a signal and outputs a FBANK SPro data file. For each frame in the signal, a 512-FFT is performed. A triangular filter-bank is applied to the module of the FFT and the log-energy in each filter is returned. If the energy is less than 1.0, the returned value is 0.0. The central frequencies of the filters are equally spaced along the transformed frequency scale if the *-a* switch is used (along the linear scale if $\alpha = 0.0$) or along the MEL scale if the *-m* switch is used. Option *-n num* sets the number of filters in the filter-bank (default is 24). The MEL scale is defined by

$$Mel(f) = 2595 \log\left(1 + \frac{f}{700}\right) \quad .$$

When band-limiting is specified (*-i* and *-u* switches), the filter-bank is only applied between the specified lower and upper bound frequencies.

3.4 sfbcep

Usage: sfbcep [options] infile outfile

Filter-bank based cepstral analysis of a speech signal. This program takes as input a signal and outputs a FBCEPSTRA SPro data file. For each frame in the signal, a filter-bank analysis is performed as in *sfbank* (the *-n*, *-a* and *-m* options are the same as in *sfbank*). A discrete cosine transform is applied to the output of the filter-bank according to:

$$c_i = \frac{1}{\sqrt{2\pi}} \sum_{j=1}^{N} \cos\left(\frac{i(j-0.5)\pi}{N}\right) \qquad i = 1, \ldots, P \quad ,$$

where m_j denotes the output of the j'th filter. The *-p num* option sets the order of the DCT (default is 16). The *-z* option turns on the long term cepstral mean subtraction and *-r num* sets the liftering value. The lifter is defined by

$$lifter[i] = 1 + \frac{l}{2} \sin\left(\frac{i\pi}{l}\right) \qquad i = 1, \ldots, P \ .$$

Band-limiting is applied as for *sfbank* before the discrete cosine transform.

3.5 slpc

Usage: slpc [options] sigfile datafile

Linear predictive analysis of a speech signal. This program takes as input a signal and outputs a LPC, PARCOR or LAR SPro data file. For each frame of the input signal, the generalized LPC analysis of order p is performed using the Levinson-Robinson-Durbin procedure to solve the Yule-Walker equations (implemented as in [5]). Option *-p num* controls the order of the LPC analysis. Actually, this procedure computes the reflexion coefficients which are converted into prediction coefficients or log-area ratio. The default behavior is to output the prediction coefficients but option *-k* (resp. *-g*) enables to save reflexion (resp. log-area ratio) coefficients to the output file.
Warning: the pre-emphasis coefficient is set using the *-e* switch and the log energy is appended using the *-E* switch[3].

3.6 slpcep

Usage: slpcep [options] sigfile datafile

LPC-based cepstral analysis of a speech signal. This program takes as input a signal and outputs a LPCEPSTRA SPro data file. For each frame in the input signal, the prediction coefficients a_i are computed as in *slpc*. Option *-p num* sets the LPC analysis order and the default is 16. The cepstral coefficients are then computed using the following recursion:

$$c_i = -a_i + \frac{1}{i} \sum_{j=1} i - 1 \left((i-j) a_j c_{i-j}\right) \text{ for } i = 1, \ldots, N$$

where P is the order of the cepstral analysis defined using the *-n num* option (default is 16). As for *sfbcep*, the *-z* option turns on the long term cepstral mean subtraction and *-r num* sets the liftering value.

4 Programmer guide

All the type and constant definitions, the function prototypes are defined in the header file *spro.h*[4]. All the functions return OK (*i.e. 0*) if no error occurs.

[3] I must admit this is a little bit confusing but that's the way it is!

[4] Actually, the header file is, to my mind, the best programmer manual one can ever find

4.1 Type definitions

Two main types are defined and used throughout the library. The first one is used for signals while the second one for data (*i.e.* parameterized signals). A signal is stored in a structure defined as:

```
typedef struct {
        unsigned long n;        /* Number of samples         */
        sample_t *s;            /* List of samples           */
} sig_t;
```

Data are stored in structures defined as:

```
typedef struct {
        datakind_t kind;        /* Sample kind               */
        unsigned long dim;      /* Vector dimension          */
        unsigned long n;        /* Number of samples         */
        flag_t flag;            /* Qualifiers                */
        param_t *s;             /* Pointer to the samples    */
} data_t;
```

Features are often returned from the base functions through a vector defined as:

```
typedef struct {
        unsigned long n;        /* Number of elements        */
        param_t *data;          /* List of elements          */
} vector_t;
```

4.2 I/O functions

The following functions are used to load a signal from a file:

- *err_t ReadSig(sig_t **,unsigned long,FILE *);*
- *err_t ReadSphereSig(sig_t **,float *,FILE *);*

The second argument of *ReadSig()* is the number of samples to be read. The second argument of *ReadSphereSig()* is used to returned the sample rate value found in the SPHERE header.

The following functions are used to load data from a file:

- *err_t ReadSPData(data_t **,FILE *);*
- *err_t LoadSPData(data_t **,flag_t,FILE *);*

The difference between those two functions is that the later also makes conversions. Actually, the second argument of *LoadSPData()* is the qualifier that the returned data should have. Therefore, *LoadSPData()* first invoke *ReadSPData()* and then convert the data using the *ConvertData()* routine. An error is returned if the conversion is impossible. The conversion function (*err_t ConvertData(data_t **,data_t *,flag_t)*), is in charge of simple conversions such as subtracting the long term mean, adding the Δ and $\Delta\Delta$ coefficients ...

The output functions for signals and data are:

- *err_t WriteSig(sig_t *,FILE *);*
- *err_t WriteSPData(data_t *,FILE *);*

4.3 LPC-related functions

- *err_t ARGModel(sig_t *sig,float a,datakind_t kind,int p,vector_t *v,float *s)*
  Perform generalized linear predictive analysis of order *p* on signal *sig*, using the spectral resolution defined by *a*. Output can be The energy of the filter $\frac{\sigma}{A(z)}$ is returned in *s*. Note that *sig* is usually a frame.

- *err_t SigARInverse(sig_t *y,vector_t *v,datakind_t kind,float sigma,int st,int en,sig_t *x)*
  Apply inverse filter $A(z)/\sigma$ to signal x between samples *st* and *en* and stores the result in *y*. *y* is NOT allocated by the function.

- *err_t SigARFilter(sig_t *y,vector_t *v,datakind_t kind,float sigma,int st,int en,sig_t *x)*
  Apply direct filter $\sigma/A(z)$ to signal x between samples *st* and *en* and stores the result in *y*. *y* is NOT allocated by the function.

- *err_t ARConvert(vector_t *v1,datakind_t t1,datakind_t t2, vector_t *v2)*
  LAR ↔ PARCOR ↔ LPC conversions. *v1* contains the source coefficients of type *t1* which are converted in *t2* type coefficients returned in vector *v2*. *v2* is NOT allocated in the function.

- *err_t LPCepstrum(vector_t *a,int p,vector_t *c)*
  computes p cepstral coefficients returned in vector c from the prediction coefficients of vector a.

4.4 FFT-related functions

- *err_t FFTransform(sig_t *x,unisgned long npts,vector_t *modul,vector_t *phase)*
  Performs the FFT of signal x using *npts* points and returns the phase in *phase* and the module in vector *modul*. Vectors *phase* and *modul* must be allocated before calling this function and must be able to contain (at least) *npts*/2 points. Either *phase* or *modul* can be NULL if phase or module is not to be computed.

- *err_t LogFilterBank(sig_t *x,float alpha,int nfilt,float f_min,float f_max,vector_t *v)*
  Applies a triangular filter bank of *nfilt* filters to signal x using the spectral resolution defined by *alpha* (*alpha*=0 means equal resolution). The frequency range (in the original domain) can be specified by the *f_min* and *f_max* variable. These frequencies must be given as normalized frequency (*i.e.* $\in [0, 0.5]$) and the whole band is considered if both frequencies are equal. The log energy in each filter is returned in vector v. v is NOT allocated in the function.

- *err_t LogMelFilterBank(sig_t *x,float freq,int nfilt,float f_min,float f_max,vector_t *v)*
  Similar to *LogFilterBank()* except that Mel frequency warping is used instead of alpha-warping. *freq* is the sample rate of the signal. The frequency range (*f_min* and *f_max*) is given in Hertz rather than using normalized frequencies.

- *err_t DCTransform(vector_t *v1,int p,vector_t *v2)*
  Computes p cepstral coefficients returned in vector *v2* from the filter-bank output in vector *v1*. *v2* is NOT allocated in the function.

4.5 Miscellaneous functions

4.5.1 Related to signals

- *err_t SigGetFrame(sig_t *sig,int nth,int l,int d,sig_t *frame)*
  Returns in structure *frame* the *nth* frame in signal *sig* assuming that the frame length is *l* samples and the frame decay *d*. First frame is frame 1. The memory for frame is NOT allocated in this function.

- *float SigNorm(sig_t *sig,bool_t flag)*
  Returns the energy in signal *sig*. If flag is true, then the signal energy is normalized to 1.

- *err_t SigWeight(sig_t *sig,win_t window)*
  Apply the specified window to the signal.

4.6 Related to data

- *unsigned long DataVecSize(data_t *data)*
  Returns the real length of the feature vectors.

- *param_t *DataGetVec(data_t *data,int n,unsigned long nelem)*
  Returns a pointer to vector n in the data. The third argument is the real feature vector length to avoid recomputing it at each invocation of this function. Vectors are numbered starting with index 0 and NULL is returned if n has an invalid value.

- *DataZeroMean(data_t *data)*
  Suppress the long term mean of the static coefficients in *data*.

- *void CepLifter(vector_t *v,int l)*
  Apply cepstral liftering to coefficients in *v*.

- *void ScaleEnergy(data_t *p,float s)*
  Scale energy according to s.

5 History

5.1 Changes from SPro 3.0 to 3.1

The organization changed from SPro 3.0. The former spro_util.* codes are now part of the libgg 2.1 library and SPro 3.1 requires that library. This library also provides an error reporting function as well as some standard error codes.

The FFT algorithm was changed to make it faster. It was previously based on Nielsen's implementation of the algorithm. The new version was adapted from the real FFT function described in the Numerical Recipes in C[5]. In SPro 3.0, the FFT was done on 1024 points and it is now done on 512 points. Current version of sfbank is about 5 times faster than the previous one.

The computation of the Δ coefficients slightly changed at the boundary (the boundary frames are now repeated instead of being avoided).

Known bugs have been corrected, principally concerning the cepstral mean subtraction and error reporting is now better. Note that for a better error reporting, many function prototypes changed so that some tools developed with the previous version may be adapted before they are compatible with SPro 3.1. The order of the enumeration for the datakind_t also changed so that when using *slist* with old data file, the data kind printed on screen may not be the correct one.

The SPHERE compatibility is also a new feature of SPro 3.1.

5.2 Changes from SPro 3.1 to 3.2

SPro 3.2 is fully compatible with version 3.1 and only minor corrections and add-ons were made[6]. The list of modifications is:

- long option names were added to all runtimes (now require version 2.2 of libgg),
- tracing has been simplified,
- energy scale factor option added to related tools,
- frequency range limiting in FFT based spectral analysis, and
- the bug in the computation of the $\Delta\Delta$ energy has been corrected ($\Delta\Delta$E was previously computed as ΔE).

References

[1] Christian Chouzenoux. *Analyse spectrale à résolution variable. Application au signal de parole.* PhD thesis, Ecole Nationale Superieure des Télécommunications, 1982.

[5] See for example the following URL: http://cfatab.harvard.edu/nr/bookc.html

[6] Compatibility is only true for the runtimes and for the algorithms but two function prototypes were changed in *spro.h*.

[2] G. Gravier, C. Mokbel, and G. Chollet. Model dependent spectral representations for speaker recognition. In *EuroSpeech*, volume 5, pages 2299–2302, 1997.

[3] C. Mokbel, G. Gravier, and G. Chollet. Optimal state dependent spectral representation for HMM modeling: a new theoretical framework. In *EuroSpeech*, volume 3, pages 1351–1354, 1997.

[4] Steve Young. *The HTK book v2.0.*

[5] René Boite and Murat Kunt. *Traitement de la parole.* Presse Polytechniques Romandes, 1982.