RASTA_PLP

David Burton

Last updated: 9/22/98

An ESPS program for robust speech-recognition feature computation

Introduction

In this application note we explain how to use the ESPS program rasta_plp to compute features for use in automatic speech recognition systems, such as can be built by using the hidden Markov model tool kit, HTK [1]. We also describe how to use rasta_plp to compute spectrograms that correspond to the auditory-based processing used by the rasta_plp program. Finally, we provide an introduction to the signal processing techniques known as perceptual linear prediction (PLP) and relative spectra (RASTA), and we list relevant literature references. Much of the background information and description found here is based on the material contained in references [2,3]

The program rasta_plp wraps ESPS file I/O around a program developed primarily by Hynek Hermansky and Nelson Morgan. The ESPS program rasta_plp supports both PLP [2] and RASTA [3] processing, and you control the program via command line options 1. This allows you to exploit existing ESPS programs for pre- and post-processing of the data and to view and manipulate PLP-related parameters interactively via ESPS/waves+. In addi-

1. If you are familiar with the original version of the RASTA program, note that it and the ESPS version of rasta_plp have several command-line option-letter differences.

Entropic Research Laboratory, Inc.
RASTA_PLP

EnSig now offers both PLP- and RASTA-based spectrogram and parameter computation.

rasta_plp addresses three important problems in speech signal processing for automatic speech recognition:
- robust feature selection;
- mismatched input analog-channels, such as are introduced by using different microphones, telephone lines, etc.;
- mismatched additive background noise environments.

In particular, PLP addresses robust feature extraction, log RASTA addresses mismatched channel distortion, and J RASTA addresses mismatched additive noise distortion. Each of these techniques is discussed in its own section below.

This Application Note should be read together with the ESPS manual pages for the programs rasta_plp and jMap. Those manual pages give a brief overview of the programs and full descriptions of all options. There are other ESPS programs that are useful for computing features for speech recognition and for normalizing data to remove environment-induced distortions. See the manual pages for the following programs for details: acf, toep_solv, filter, and cepanal.

Perceptual Linear Prediction

As mentioned above, PLP parameters are robust features when used in automatic speech recognition experiments. This section provides background information on:
- the signal processing used to compute PLP features,
- why doing this processing is useful, and
- how the processing is implemented in rasta_plp.

Finally, this section shows examples of using rasta_plp to compute features for use with a speech recognizer.
RASTA_PLP

What Are PLP Parameters?
PLP parameters are the coefficients that result from standard all-pole modeling [7], or linear predictive analysis, of a specially modified, short-term speech spectrum. In PLP the speech spectrum is modified by a set of transformations that are based on models of the human auditory system. To be more precise, the following three concepts from the psychophysics of hearing are applied to derive an auditory spectrum estimate:

• critical-band spectral-resolution [4],
• the equal-loudness hearing curve [5], and
• the intensity-loudness power law of hearing [6].

Once the auditory-like spectrum is estimated, it is converted to autocorrelation values by doing a Fourier transform. The resulting autocorrelations are used as input to a standard linear predictive analysis routine, and its output is perceptually-based linear prediction coefficients. Typically, these coefficients are then converted to cepstral coefficients via a standard recursion [7].

Why Use PLP Parameters?
Linear prediction coefficients (LPC) have a long history of use in automatic speech recognition. These coefficients are often used because they approximate well the high-energy regions of the speech spectrum while simultaneously smoothing out the fine harmonic structure, which is often characteristic of the individual but not of the underlying linguistic unit. LPC, however, approximates the speech spectrum equally well at all frequencies, and this representation is contrary to known principles of human hearing.

The spectral resolution of human hearing is roughly linear up to 800 or 1000 Hz, but it decreases with increasing frequency above this linear range. PLP incorporates critical-band spectral-resolution into its spectrum estimate by remapping the frequency axis to the Bark scale and integrating the energy in the critical bands to produce a critical-band spectrum approximation.

At conversational speech levels, human hearing is more sensitive to the middle frequency range of the audible spectrum. PLP incorporates the effect of this phenomenon by multiplying the critical-band spectrum by an equal-
RASTA_PLP

loudness curve that suppresses both the low- and high-frequency regions relative to the midrange from 400 to 1200 Hz.

In addition, there is a nonlinear relationship between the intensity of sound and the perceived loudness. PLP approximates the power-law of hearing by using a cube-root amplitude-compression of the loudness-equalized critical-band spectrum estimate.

The complete psychoacoustically derived spectrum, produced by combining all 3 of the above described transformations, has less detail and a smaller dynamic range than the original spectrum. Because of this, it can be modeled well by a low-order all-pole model, which is effective in suppressing speaker-specific details of the spectrum. In addition, the PLP order is smaller than is typically needed by LPC-based speech recognition systems, and this lower analysis order results in “better” estimates of recognition parameters for a given amount of training data.
How is PLP Implemented?

Each box in Figure 1 represents a separate signal processing step in the `rasta_plp` program during the computation of PLP coefficients. Some steps have user-settable controls. The items listed beneath a box represent `rasta_plp` command line options that affect the signal processing within that block. For example, the “Frame the Data” block has three command line options that affect its output. The `-1` option controls the amount of data in each frame, etc. See the `rasta_plp` man page for a description of the parameters.

There are three possible output points from the `rasta_plp` program, indicated by the solid circles. Only one output at a time is possible during pro-
gram execution, however, and PLP cepstral coefficients are output by default. The command line options to turn on other outputs are shown in Figure 1.

How Are PLP Parameters Computed?

Bark-Warped Critical-Band Spectrogram. The following command produces a critical-band spectrogram. This spectrogram file can be viewed by using xwaves.

% rasta_plp -R -S2 -l8 speech.sd speech.cbsgram

Figure 2 shows an xspectrum display of a spectral slice from the portion of the word “usually” with standard FFT- and LPC-based spectrum overlays. The FFT overlay is the jagged one, and the LPC overlay is the smooth one. Note that the first critical-band filter value is set to 0. This is done because the filter bandwidth extends into the negative frequency region. Similarly, the last filter output value at 4000 Hz is set to 0.

FIGURE 2. Critical-band spectrum with FFT and LPC overlays
RASTA PLP

Complete Auditory-Processed Spectrogram. The following command produces a spectrogram that results from processing the original spectrum with all three auditory-like transforms.

`% rasta-plp -S2 -l8 -P speech.sd speech.audsgram`

Figure 3 shows a spectral slice from the file `speech.audsgram`. The corresponding spectral slice from the critical-band spectrogram is shown as an overlay. Note that in the full auditory-processed output, the spectrum value for the first and last critical-band filter is set equal to the one immediately adjacent to it. This is done rather than actually computing the extreme low and high frequency bands, since the replaced spectral values fall outside of the band that is typically well-estimated, and are generally not significant for phonetic identity.

**FIGURE 3. Complete auditory-processed spectrum with critical-band spectrum overlay**

PLP-Based Cepstral Coefficients. The following command produces PLP cepstral coefficients:

`% rasta_plp -S2 -l8 -e 0.0 speech.sd speech.plp`

Note that cepstral liftering is turned off by setting the liftering exponent to 0.0 via the `-e` option.
RASTA PLP

PLP-Based Spectrum. As mentioned above, the PLP spectrum estimate is a smoothed version of the full auditory-processed spectrum. The smoothing is accomplished by fitting an autoregressive model to the auditory spectrum estimate. Here we show the ESPS command used to convert the set of autoregressive coefficients (in this case cepstral coefficients) into an equivalent spectrum:

```
% me_spec -n17 -G -P PlpParams \\
speech.plp speech.plp.fspec
```

where the file PlpParams is an ESPS parameter file containing the following:

```
string spec_param_field = "plp[1:5]";
string spec_rep = "CEP";
float power = 100000.;
float samp_freq = 8000.;
```

For details on the particulars of ESPS parameter files refer to the Application Note “Parameter and Common Files”.

In the above command, the input cepstral coefficients are from the same data frame as used in computing the plots shown in Figure 3. Figure 4 shows the resulting spectrum estimate with the corresponding full auditory-processed filter-bank spectrum estimate as an overlay. Note that the PLP-based spectrum is smoother than the corresponding auditory-processed filter-bank output and has the same gross features (two peaks), but the PLP-based spectrum is shifted to the right relative to the filter-bank output spectrum. This shift is caused by the application of a standard (uniformly spaced) inverse discrete Fourier transform (see Figure 1 on page 5) on the non-uniformly spaced filter-bank output, prior to the LPC analysis.
The plot in Figure 4 is the result from using no cepstral liftering. By default, rasta_plp uses a cepstral liftering value of 0.6 in PLP processing. Figure 5 shows the PLP-based spectrum estimate after applying a cepstral lifter of 0.6 with the non-liftered spectral estimate from Figure 4 as an overlay. As can be seen, the cepstral liftering sharpens the frequency peaks considerably.

Entropic Research Laboratory, Inc.
RASTA PLP

How is PLP Used in a Recognition System?

`rasta_plp` produces PLP feature vectors. These feature vectors can be used directly in the training and testing of a vector-quantization-based, dynamic-time-warping-based, or hidden-Markov-model-based speech recognition system. See [9] for information on parameter values that provide “good” features for different tasks.

Log-RASTA Filtering

A major cause of problems in fielded speech recognition systems is the mismatch between the conditions used to record the speech training data and the conditions under which the data to be recognized is recorded. Sometimes the differences are approximated fairly well by a linear filter, resulting in a multiplicative change in the frequency domain. Examples of such phenomena are a change in microphone or handset, or in telephone lines (e.g., training on the public-switched network and testing on an internal PABX connection). These types of differences are known as “input channel mismatch,” and log-RASTA addresses these differences.

What is Log-RASTA Filtering?

The term RASTA comes from the words *RelAtive SpecTrA*. The RASTA technique applies a bandpass filter to each spectral component in the critical-band spectrum estimate. This filtering emphasizes frame-to-frame spectral changes that occur between the rates of 1 to 10 Hz. Before applying the bandpass filter, log-RASTA takes the natural logarithm of each spectral component. This logarithm converts multiplicative distortions in the frequency domain into an additive distortion, which can be filtered. Conversion to the log-spectrum domain is a common approach used in signal deconvolution problems.

Why Use Log-RASTA Filtering?

The rate of change of nonlinguistic components of speech and background noise environments often lie outside the typical rate-of-change of vocal-tract shapes in conversational speech. Also, informal studies showed that human
hearing seems relatively insensitive to slowly varying stimuli [2]. The basic idea of RASTA filtering is to exploit these phenomena by suppressing constant and slowly varying elements in each spectral component of the short-term auditory-like spectrum prior to computation of the linear prediction coefficients.

In addition, speaker-generated high-frequency modulations of the spectral content convey little phonetic information and thus represent differences that are not crucial to the automatic speech recognition process. The high-frequency cut-off part of the bandpass filter attenuates (and thus normalizes across speakers) these high frequency variations, as well as any channel/processing induced spectral changes that occur more rapidly than is seen in typical conversational speech.

Thus RASTA highpass filtering removes slowly varying components in each element of the filter-bank output, such as introduced by communication channels, and RASTA lowpass filtering removes rapidly changing components, typical of changes that are not phonetically important.

**How is Log-RASTA Filtering Implemented?**

*rasta_plp* implements log-RASTA filtering by inserting the steps illustrated in Figure 6 after the critical-band spectrum computation and before the equal-loudness weighting (refer to Figure 1 on page 5).
RASTA PLP

**FIGURE 6. Log RASTA filtering**

The RASTA bandpass filter has the following system function:

\[ H(z) = \frac{(z^4 + 0.2 + 0.1z^{-1} - 0.1z^{-3} - 0.2z^{-4})}{1 - 0.94z^{-1}} \]

and the corresponding frequency response shown on a log-log scale:

**FIGURE 7. RASTA filter frequency response**
RASTA_PLP

The `--p` option on `rasta_plp` allows you to change the pole position of the bandpass filter from the default value of 0.94 (but the filter cannot be changed otherwise without modifying the program source code). Also note that when RASTA filtering is being done, the `rasta_plp` output produced by using the `-R` option does not include the inverse nonlinear-mapping of the data. Normally, this output is used only when doing adaptive J RASTA filtering — see the section “J-RASTA Filtering” on page 15.

Finally, please note two things. The filter characteristics were optimized on a particular task and for a particular analysis frame rate (100 frames/sec.). For this frame rate, Figure 7 shows a passband between about 1 and 10 Hz. Small changes in the frame rate should not degrade seriously the preferred frequency response, but ideally the filter coefficients should change with frame rate to keep the response constant. Secondly, the filter is an IIR filter, and it has significant memory. RASTA output depends on where the analysis starts, and thus the preceding context.

How Are Log-RASTA Parameters Computed?

Log-RASTA Filtered Auditory Spectrogram. The following command computes a spectrogram that contains the log-RASTA filtered auditory-based spectrum estimate:

```
% rasta_plp --L --P -S2 -l8 speech.sd \
  speech.spectrogram
```

Figure 8 shows the log-RASTA filtered and the plain auditory-processed spectrograms for comparison. Note both the enhanced onsets and the fading-away steady-vowel formants in the RASTA processed spectrogram.
RASTA_PLP

FIGURE 8. Log-RASTA filtered and complete auditory-processed spectrograms

Log-RASTA Filtered Cepstral-Coefficients. The following command produces log-RASTA filtered cepstral coefficients:

```
% rasta_plp -L speech.sd speech.rplp
```

How is Log-RASTA-PLP Used in a Recognition System?

You train and test speech recognition systems with these parameters by using exactly the same methods as those used with PLP parameters. The only difference is when to use these features. These features are best used when there is a mismatch in the analog input channel between the development and fielded systems.
J-RASTA Filtering

Typically the mismatch between the environments in development and fielded systems includes more than just the input analog-channel mismatch. Often it also includes a mismatch in the background noise, either in level, spectral shape, or both. J-RASTA was developed to address the combined problem of convolutional and additive noise.

What is J-RASTA Filtering?

J-RASTA filtering is a noise reduction technique. In J-RASTA filtering, the logarithmic and exponential transformations of the log-RASTA processing technique are replaced by nonlinear mapping functions that are noise-signal dependent. This modification allows the bandpass filter to work in the linear domain on additive-noise dominated parts of the signal and in the log domain on speech-dominated parts of the signal.

Why Use J-RASTA Filtering?

J-RASTA filtering helps compensate for both additive and convolutional noise. It has the benefits of log-RASTA on frequency ranges of the signal that have high signal-to-noise ratios (SNR). On frequency regions of the signal that have low SNR, J-RASTA filters both slowly varying and rapidly varying additive noise. There are two types of J-RASTA filtering supported by rasta_plp: constant-J and adaptive-J. When the noise level is known ahead of time and it is fairly constant, use constant-J processing. When the noise characteristics are changing, use adaptive-J processing, which uses an updated frame-by-frame estimate of the noise to set the J value appropriately for each input frame.

How is J-RASTA Implemented?

Both J-RASTA processing approaches insert additional signal processing steps between the critical-band spectrum computation and the equal-loudness weighting of the PLP computation. (See Figure 1.)

Constant-J RASTA Filtering. The constant-J filtering implementation is shown in Figure 9:
As seen in Figure 9 above, constant-J processing is a simple modification of the standard log-RASTA processing (Figure 7 on page 12). The bandpass filter used in constant-J RASTA is identical to the one used in log-RASTA. The difference in the two methods is in the nonlinearity used before and after the bandpass filtering. In constant-J RASTA, a noise-level dependent nonlinear-mapping is used that has the following form:

\[ Y = \log(1 + J \times X) \]

where \( J \) is a noise-dependent value and \( X \) is a critical-band-spectrum value. After filtering, the spectral trajectory values are mapped back from the log domain by the following inverse mapping:

\[ X = \frac{e^Y}{J} \]

Note that the exact form of the inverse nonlinear mapping is not used because it can yield a negative power-spectral value, which, of course, is physically impossible. The inverse mapping shown above is equal to the exact inverse plus \( 1/J \).
RASTA_PLP

Adaptive-J RASTA Filtering. The adaptive-J filtering implementation is shown in Figure 10:

FIGURE 10. Adaptive-J RASTA filtering

In adaptive-J processing, the form of the nonlinear mapping ($\log(1 + J \times X)$) used on each individual spectral component is identical to that used in constant-J processing, however, the J-value used for each frame can change.

You prepare for adaptive-J processing by computing, off-line and ahead of time, linear mapping coefficients for each of a set of “representative” J-values. By representative, we mean values that correspond to the expected range of noise conditions in the fielded environment. The linear mapping coefficients, corresponding to a specific J-value, map the bandpass filtered critical-band spectrum to values expected when processed by a pre-determined “reference” J value. The reference value is the one used when training the recognizer. The program \texttt{jMap} calculates mapping coefficients for use with \texttt{rasta_plp}.

\textit{Entropic Research Laboratory, Inc.}
RASTA PLP

Adaptive-J processing first computes a mean noise-energy estimate for each frame by using a technique based on the method described by Hirsch [8]. Each frame’s noise estimate is compared with the set of representative J-values, and the closest is chosen. This quantized J-value is used in the nonlinear mapping function prior to bandpass filtering (see Figure 10).

The bandpass filtered output is then transformed via the linear mapping coefficients associated with the quantized J-value for this frame to values that correspond to the previously determined reference J-value. Subsequently, these values are processed by the inverse nonlinear mapping defined by the reference J-value. The output from the inverse nonlinearity is used in the rest of the standard PLP processing.

What Problems Are Caused by Using J RASTA?

Both constant-J and adaptive-J processing are noise-signal dependent, so accurate estimation of noise characteristics is important to the success of these methods. For constant-J processing, if you select the wrong J-value or the noise characteristics change during the experiments, performance can be adversely affected. For adaptive-J processing, the testing must contend with the variability introduced by the J-value changing over time. The spectral mapping to the reference J-value tries to reduce the variability caused by the changing J-values, but, in general, you are better off if your environment allows you to use constant-J processing.

The noise estimation procedure used in adaptive-J processing requires at least 100 msec. of speech-free signal at the beginning of the data. Failure to provide this results in poor noise estimates and bad parameter estimates.

How Are J-RASTA Parameters Computed?

Constant-J RASTA Filtered Coefficients. The following command uses a J-value of $3 \times 10^{-9}$ and produces 8 output cepstral coefficients that are computed from frames that are 25 msec. long and that overlap by 12.5 msec.:%

```bash
rasta_plp -J -C -j 3e-9 -n 8 -l 25 -S 12.5 \speech.sd speech.cjrplp
```

J-RASTA Filtering
RASTA_PLP

The -C option sets rasta_plp in the constant-J mode when J RASTA processing is enabled via the -J option.

Adaptive-J RASTA Filtered Coefficients. The -j option turns on adaptive-J processing in rasta_plp, but you must provide a J-mapping file via the -f option. You create a mapping file by using the jMap program on a set of training data. The following command takes an input ASCII file containing paths to speech data files, a set of representative J-values, and a reference, or target, J-value as input.

```
% jMap -i fileList -l "1.0e-9 1.0e-8 1.0e-7" \
     -j 3.0e-6 mapFile
```

In the above command, we specified the reference J value as $3 \times 10^{-6}$ and a set of linear mapping coefficients is created for each representative J-value enclosed in double quotes ("). These mapping sets are written to the output file mapFile.

Each mapping coefficient set is associated with a representative J value. When a J value is used to nonlinearly map the spectral coefficients prior to bandpass filtering, the J-value’s associated linear mapping function transforms the bandpass filter outputs to values that correspond to processing the data with the reference J value — the value used to train the recognition system.

After creating the mapping file, the following command does adaptive-J processing and produces cepstral coefficients:

```
% rasta_plp -J -f mapFile speech.sd speech.ajrplp
```

The default values for many rasta_plp options are used in the above two commands. Typically, you need to ensure that rasta_plp processing option values are identical to those used by jMap when creating the file containing the linear mappings.
How is J-RASTA-PLP Used in a Recognition System?

As mentioned above, the appropriate J-value to use in J-RASTA is noise-signal dependent. More precisely, experience has shown that the J-value appears optimally set when it is inversely proportional to the mean critical-band noise energy ($N_{energy}$). Experiments [3] showed $1/(3 \times N_{energy})$ as a good choice, but the best choice for your conditions may vary.

Using Constant-J RASTA. This method is appropriate when the background noise is roughly stationary. If this is true, measure the mean noise-energy across the critical-band outputs and use it to set the J value via the $-j$ option for testing the recognizer.

You measure the mean noise-energy by using ESPS tools. First capture the critical-band outputs by using the $-R$ option of rasta_plp. This produces a spectrogram whose values are stored in dBs. You convert these dB values, assumed to be in the file sp.cbsgram, to linear values by the following ESPS command pipeline:

```
% feafunc -f re_spec_val -f lin_spec_val -d -12 \
    -t float sp.cbsgram - | \ 
    feafunc -f lin_spec_val -f lin_spec_val | \ 
    -g 0.1429 - - \
    feafunc -f lin_spec_val -f lin_spec_val \
    -f exp10 - sp.cbvalues
```

You then compute the average critical-band values with the following command:

```
% fea_stats -M -n avgCbValues -f lin_spec_val \ 
    sp.cbvalues sp.avgCbvalues
```

Finally, you compute the mean critical-band value, over all critical bands of interest, with the following command:

```
% fea_stats -f mean\[1-15\] sp.avgCbvalues
```

The "\"s in the expression "\[1-15\]" are used to escape the brackets, which are special to csh. The expression mean\[1-15\] in the above command tells the program fea_stats to operate on a restricted range, which
excludes the critical-band filters centered at 0 Hz and at half the sampling frequency. These filters are excluded because their values were set to 0.0.

**Using Adaptive-J RASTA.** To use this method, train the recognizer by using constant-J processing with “clean” speech processed with a J value appropriate for the data. Often a reasonable J value for clean speech is $1.0 \times 10^{-6}$. Next create a set of mapping functions by finding the relationship between the filtered critical-band spectrum for a set of representative J-values and the reference-value chosen for the clean speech. The `jMap` program finds these mappings. The inputs to `jMap` are clean speech examples, the reference J value used to train the recognizer, and a set of representative J values that span the noise conditions expected in the fielded recognition environment. Typically, data from 10 or more speakers is adequate to estimate the linear mapping coefficients. `jMap` produces a mapping file containing a set of coefficients for each representative J-value specified on the command line.

Once the mapping file is created, extract speech features for recognition by using the mapping file as an input to `rasta_plp`. If the noise environment changes so much that the initial representative set of J-values are no longer appropriate, then you create a new mapping file off-line and start using it with `rasta_plp` to extract recognition features. No changes to the recognition models or templates are necessary.

**Summary**

This Application Note provides an introductory overview of the speech processing techniques called PLP and RASTA. In addition to describing the processing steps involved in computing PLP and RASTA-filtered parameters, this note explains why these parameters are useful in speech recognition and how and when they should be used. Finally, it shows how to compute these parameters by using the ESPS programs `rasta_plp` and `jMap`.

*Entropic Research Laboratory, Inc.*
**RASTA_PLP**

The table below summarizes the recommended `rasta_plp` processing technique for various training/operational environments found in speech recognition systems:

**TABLE 1. Recognizer environments and RASTA-PLP processing techniques**

<table>
<thead>
<tr>
<th>Training and operational environment</th>
<th>Processing technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matched training and testing</td>
<td>PLP or log-RASTA or constant-J RASTA</td>
</tr>
<tr>
<td>Stationary or slowly varying input-channel mismatch</td>
<td>Log RASTA</td>
</tr>
<tr>
<td>Stationary additive-noise mismatch</td>
<td>Constant-J RASTA</td>
</tr>
<tr>
<td>Stationary input-channel and stationary additive-noise mismatch</td>
<td>Constant-J RASTA</td>
</tr>
<tr>
<td>Slowly varying additive-noise mismatch</td>
<td>Adaptive-J RASTA</td>
</tr>
<tr>
<td>Slowly varying input-channel and slowly varying additive-noise mismatch</td>
<td>Adaptive-J RASTA</td>
</tr>
</tbody>
</table>

**Acknowledgments**

I would like to thank Dan Ellis, Brian Kingsbury, and Nelson Morgan from the International Computer Science Institute, Univ. of Berkeley, CA and Hynek Hermansky from the Oregon Graduate Institute, Portland OR for comments and discussions that improved both the clarity and correctness of this application note.

**References**

RASTA_PLP


