Disclaimer

- The material provided in this document is not my original work and is a summary of some one else’s work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
-

Chapter # 5: Speech Input / Output

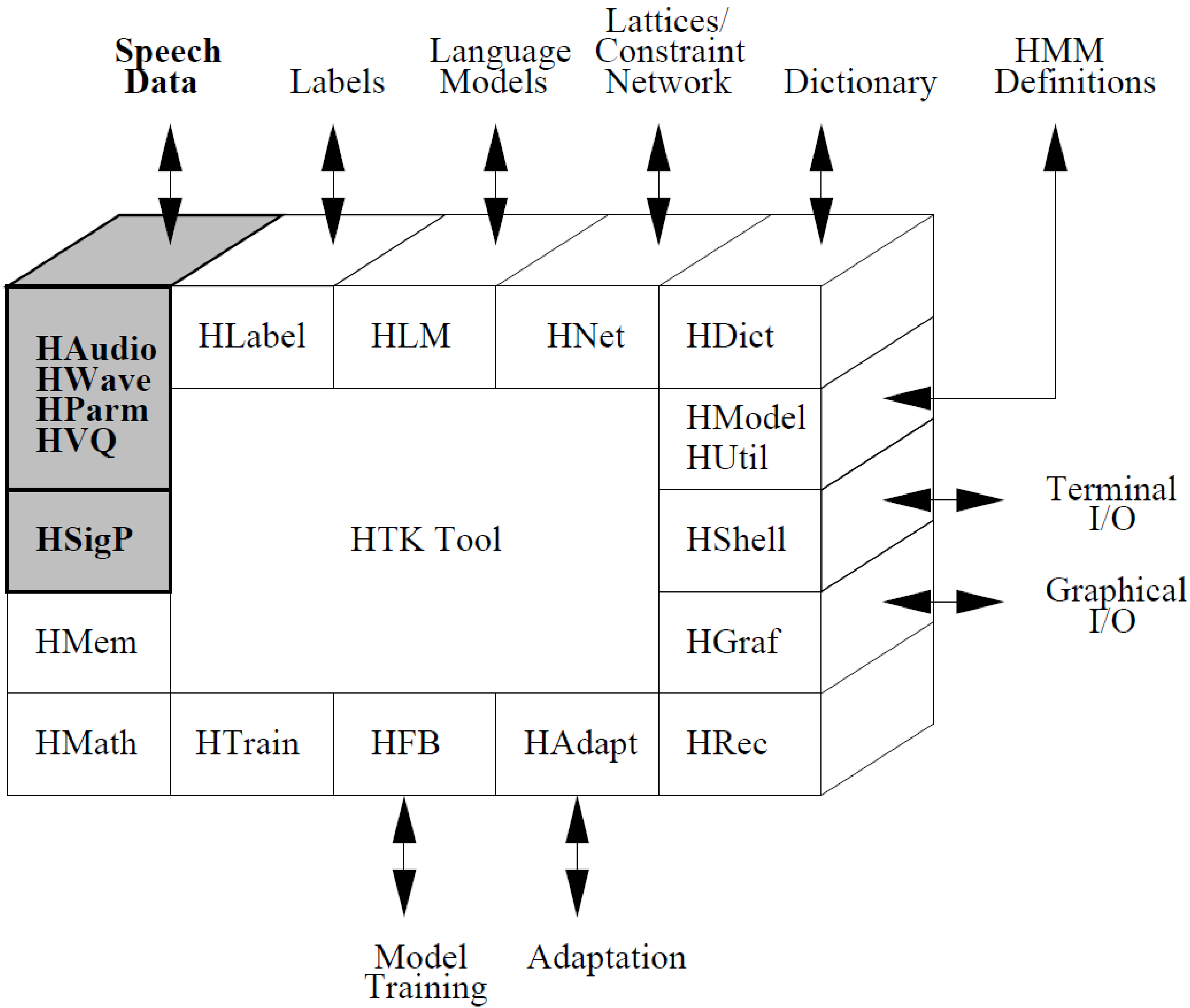
HMM TOOL KIT HTK

Outline

- ✓ Introduction
- ✓ General Mechanism
- ✓ Speech Signal Processing
- ✓ Linear Prediction Analysis
- ✓ Filterbank Analysis
- ✓ Vocal Tract Length Normalization
- ✓ Cepstral Features
- ✓ Perceptual Linear Prediction
- ✓ Energy Measures
- ✓ Delta, Acceleration and Third Differential Coefficients
- ✓ Storage of Parameter Files
- ✓ Waveform File Formats
- ✓ Direct Audio Input / Output
- ✓ Multiple Input Streams
- ✓ Vector Quantization
- ✓ Viewing Speech with Hlist
- ✓ Copying and Coding using HCopy

Introduction

- HTK provides a number of different methods for providing parameterized speech data to tools
  - Input from a previously encoded speech parameter file
  - Input from a waveform file which is encoded as part of the input processing
  - Input from an audio device which is encoded as part of the input processing

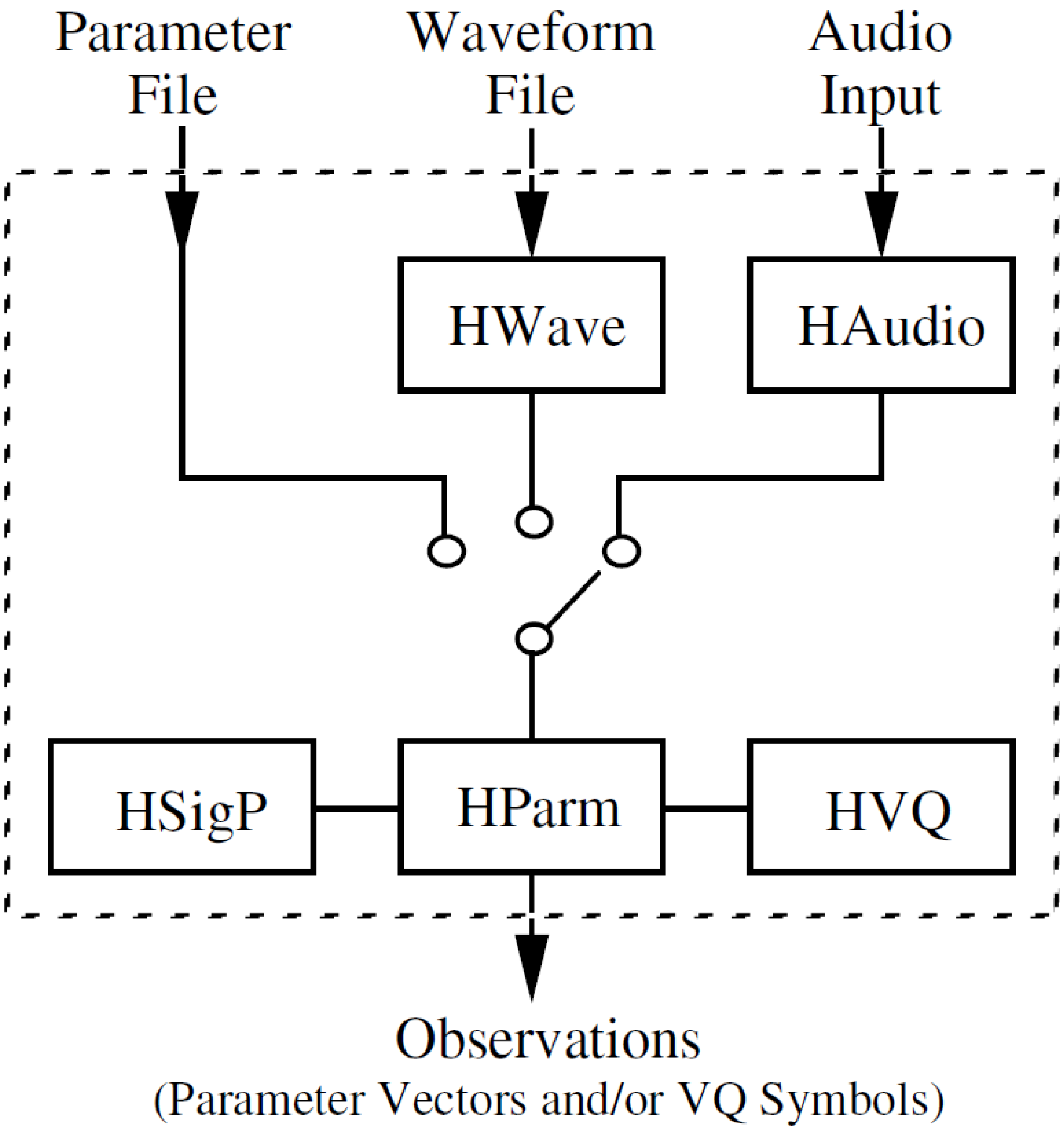


Speech Data
Labels
Language Models
Lattices/ Constraint Network
Dictionary
HMM Definitions
HAudio
HWave
HParm
HVQ
HLabel
HLM
HNet
HDict
HModel
HUtil
HSigP
HTK Tool
HShell
Terminal I/O
HMem
HGraf
Graphical I/O
HMath
HTrain
HFB
HAdapt
HRec
Model Training
Adaptation

General Mechanism

- The facilities for speech input and output in HTK are provided by five distinct modules
  - HAudio
  - Hwave
  - Hparm
  - HVQ
  - HSigP

General Mechanism



General Mechanism

```
HERest ...  s1 s2 s3 s4 ...
```

- The speech data files s1, s2, s3, . . . are inputted via the library module HParm and they must be in exactly the form needed by the tool
- If the external form of the speech data files is not in the required form, it will often be possible to convert them automatically during the input process
- To do this, configuration parameter values are specified whose function is to define exactly how the conversion should be done
- The key idea is that there is a source parameter kind and target parameter kind
- The principle function of the speech input subsystem is to convert the source parameter kind into the required target parameter kind

General Mechanism

- Parameter kinds consist of a base form to which one or more qualifiers may be attached where each qualifier consists of a single letter preceded by an underscore character

General Mechanism

WAVEFORM	simple waveform
LPC	linear prediction coefficients
LPC_D_E	LPC with energy and delta coefficients
MFCC_C	compressed mel-cepstral coefficients

```
SOURCEKIND = WAVEFORM
TARGETKIND = MFCC_E
```

```
SOURCEKIND = ANON
TARGETKIND = ANON_D
```

```
SOURCEKIND = LPC
TARGETKIND = LPREFC
```

Speech Signal Processing

- There are some simple pre-processing operations that can be applied prior to performing the actual signal analysis
  - The DC mean can be removed from the source waveform by setting the Boolean configuration parameter ZMEANSOURCE
  - It is common practice to pre-emphasize the signal by applying the first order difference equation

$$s'_n = s_n - k\, s_{n-1}$$

  - Setting USEHAMMING to true applies the following transformation to the frame

$$s'_n = \left\{ 0.54 - 0.46 \cos\left(\frac{2\pi(n-1)}{N-1} \right) \right\} s_n$$

Speech Signal Processing

```
ZMEANSOURCE = T
USEHAMMING = T
PREEMCOEF = 0.97
```

$$s'_n = \left\{0.54 - 0.46\cos\left(\frac{2\pi(n-1)}{N-1}\right)\right\} s_n$$

Speech Signal Processing

- Certain types of artificially generated waveform data can cause numerical overflows with some coding schemes
- In such cases adding a small amount of random noise to the waveform data solves the problem

$$s'_n = s_n + qRND()$$

- q= ADDDITHER; +ve value means same noise every time, -ve value means noise is random

Speech Signal Processing

- The byte-order of external speech waveform may be different to that used by the machine on which HTK is running
- HWave can perform automatic byte-swapping in order to preserve proper byte order
- If the source format is known, then HWave will make an assumption about the byte order used to create speech files in that format
- For unknown formats, proper byte order can be ensured by setting the configuration parameter BYTEORDER to VAX if the speech data was created on a little-endian machine

Linear Prediction Analysis

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-i}}$$

$$r_i = \sum_{j=1}^{N-i} s_j s_{j+i}$$

for i=1...P

$$k_j^{(i)} = k_j^{(i-1)} \qquad \text{for } j = 1, i-1$$

$$k_i^{(i)} = \left\{ r_i + \sum_{j=1}^{i-1} a_j^{(i-1)} r_{i-j} \right\} / E^{(i-1)}$$

$$E^{(i)} = (1 - k_i^{(i)} k_i^{(i)}) E^{(i-1)}$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i^{(i)} a_{i-j}^{(i-1)} \qquad \text{for } j = 1, i-1$$

$$a_i^{(i)} = -k_i^{(i)}$$

Linear Prediction Analysis

- To effect the above transformation, the target parameter kind must be set to either LPC to obtain the LP filter parameters $\{a_i\}$ or LPREFC to obtain the reflection coefficients $\{k_i\}$

```
TARGETKIND = LPREFC
LPCORDER = 12
```

- An alternative LPC-based parameterization is obtained by setting the target kind to LPCEPSTRA to generate linear prediction cepstra
- The cepstrum of a signal is computed by taking a Fourier (or similar) transform of the log spectrum

Linear Prediction Analysis

- In the case of LPC cepstra can be more efficiently computed using a simple recursion

$$c_n = -a_n - \frac{1}{n} \sum_{i=1}^{n-1} (n-i) a_i c_{n-i}$$

- The number of cepstra generated need not be the same as the number of filter coefficients, hence it is set by a separate configuration parameter called NUMCEPS

Linear Prediction Analysis

- Cepstral coefficients is that they are generally de-correlated and this allows diagonal covariances to be used in the HMMs
- The higher order cepstra are numerically quite small and this results in a very wide range of variances when going from the low to high cepstral coefficients
- It is convenient to re-scale the cepstral coefficients to have similar magnitudes
- This is done by setting the configuration parameter CEPLIFTER to some value L to lifter the cepstra

$$c'_n = \left(1 + \frac{L}{2} sin \frac{\pi n}{L}\right) c_n$$

Linear Prediction Analysis

```
TARGETKIND = LPCEPSTRA
LPCORDER = 14
NUMCEPS = 12
CEPLIFTER = 22
```

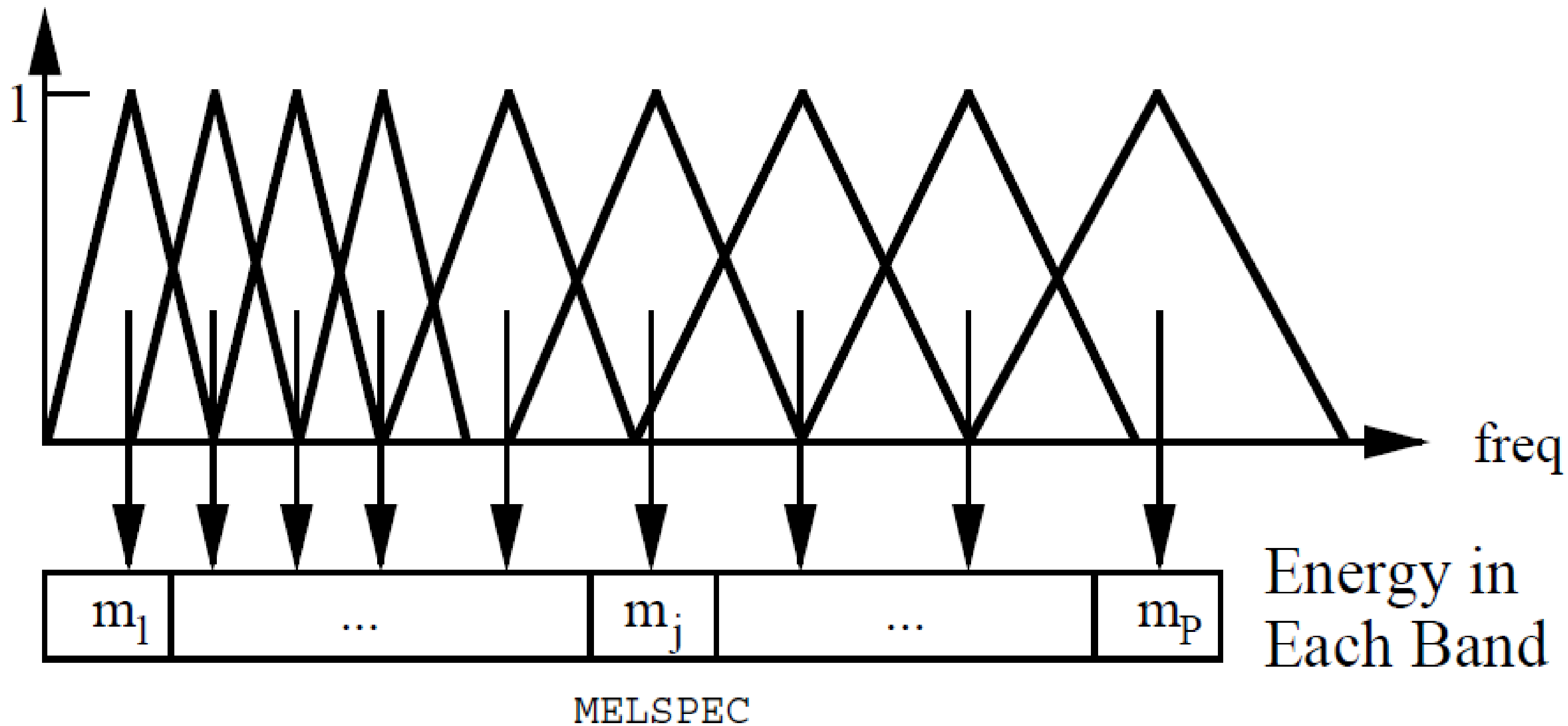
$$c'_n = \left(1 + \frac{L}{2} sin \frac{\pi n}{L}\right) c_n$$

Filterbank Analysis

- The human ear resolves frequencies non-linearly across the audio spectrum
- Filterbank analysis provides a much more straightforward route to obtaining the desired non-linear frequency resolution
- Filterbank amplitudes are highly correlated and hence, the use of a cepstral transformation is virtually mandatory
- HTK provides a simple Fourier transform based filterbank designed to give approximately equal resolution on a mel-scale

Filterbank Analysis

$$\mathrm{Mel}(f) = 2595 \log_{10}(1 + \frac{f}{700})$$



Filterbank Analysis

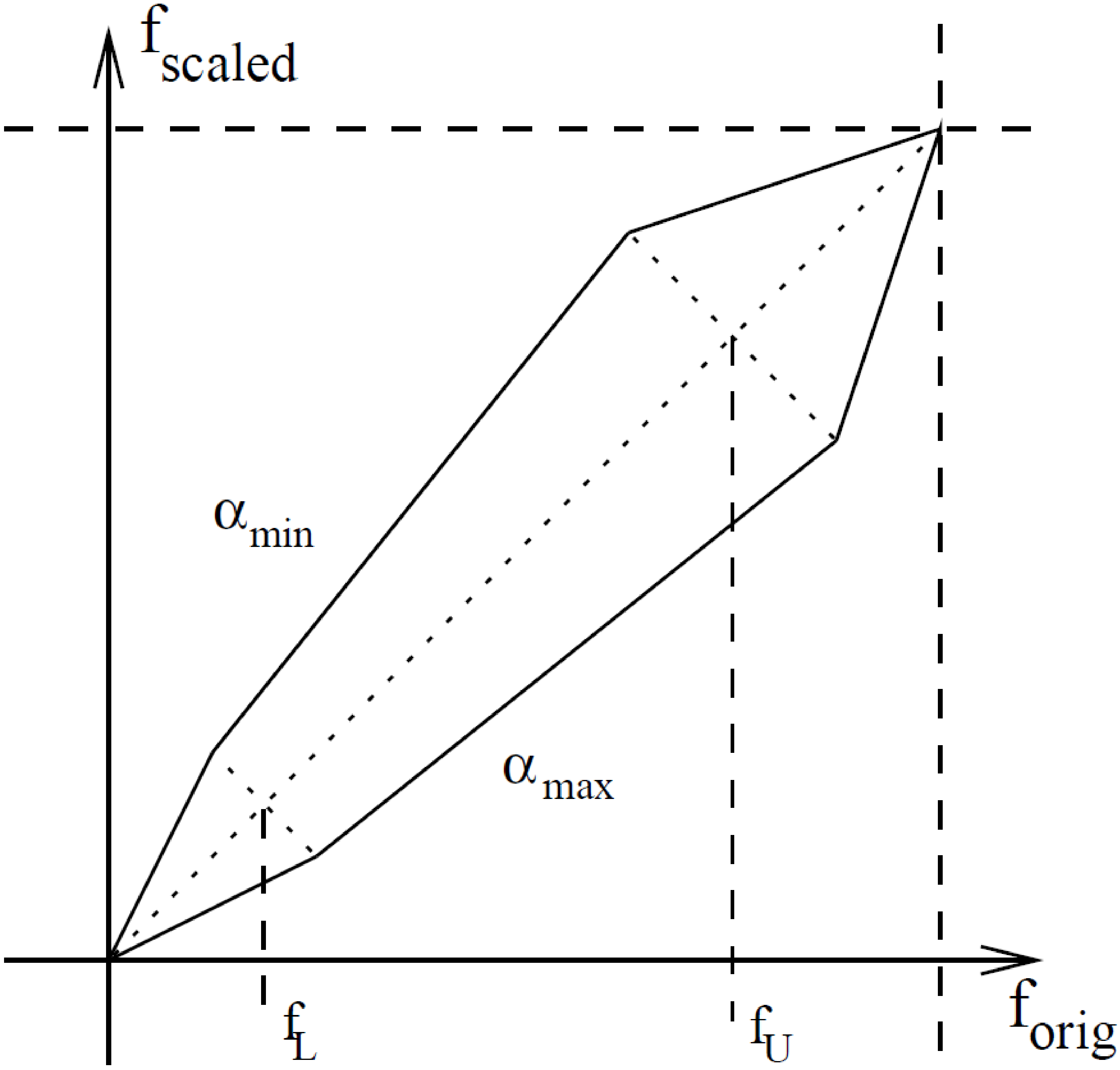
- The filters used are triangular and they are equally spaced along the mel-scale
- To implement this filterbank
  - The window of speech data is transformed using a Fourier transform and the magnitude is taken
  - Each FFT magnitude coefficient is multiplied by the corresponding filter gain and the results accumulated
  - Each bin holds a weighted sum representing the spectral magnitude in that filterbank channel
  - The Boolean configuration parameter USEPOWER can be set true to use the power rather than the magnitude of the Fourier transform in the binning process

Filterbank Analysis

- For filterbank analysis, lower and upper frequency cut-offs can be set using the configuration parameters LOFREQ and HIFREQ
- If mel-scale filterbank parameters are required directly, then the target kind should be set to MELSPEC, (FBANK for log parameters)

Vocal Tract Length Normalization

- Vocal tract length normalization (VTLN) aims to compensate for the fact that speakers have vocal tracts of different sizes
- VTLN can be implemented by warping the frequency axis in the filterbank analysis
- In HTK simple linear frequency warping is supported



f_{scaled}
α_{min}
α_{max}
f_L
f_U
f_{orig}

Cepstral Features

- Mel-Frequency Cepstral Coefficients (MFCCs) are obtained from the log filterbank amplitudes fmjg using the Discrete Cosine Transform

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right)$$

- where N is the number of filterbank channels set by the configuration parameter NUMCHANS
- The required number of cepstral coefficients is set by NUMCEPS

Cepstral Features

- MFCC give good discrimination and lend themselves to a number of manipulations
- Channel effects can be removed by subtracting the cepstral mean from all input vectors
- It compensates for long-term spectral effects such as those caused by different microphones and audio channels
- To perform this Cepstral Mean Normalization (CMN) in HTK it is only necessary to add the Z qualifier to the target parameter kind
- The mean is estimated by computing the average of each cepstral parameter across each input speech file

Cepstral Features

- To use speaker/cluster-based normalization the mean and variance estimates are computed offline before the actual recognition and stored in separate files (two files per cluster)
- The configuration variables CMEANDIR and VARSCALEDIR point to the directories where these files are stored
- To find the actual filename a second set of variables (CMEANMASK and VARSCALEMASK) has to be specified

Cepstral Features

```
CMEANDIR       = /data/eval01/plp/cmn
CMEANMASK      = %%%%%%%%%%_*
VARSCALEDIR    = /data/eval01/plp/cvn
VARSCALEMASK   = %%%%%%%%%%_*
VARSCALEFN     = /data/eval01/plp/globvar

sw1-4930-B_4930Bx-sw1_000126_000439.plp

/data/eval01/plp/cmn/sw1-4930-B
/data/eval01/plp/cvn/sw1-4930-B
```

Cepstral Features

- The file specified by VARSCALEFN contains the global target variance vector
- The variance of the data is first normalized to 1.0 based on the estimate in the appropriate file in VARSCALEDIR
- Then scaled to the target variance given in VARSCALEFN

Cepstral Features

```
<CEPSNORM> <PLP_0>
<MEAN> 13
-10.285290 -9.484871 -6.454639 ...
```

```
<CEPSNORM> <PLP_D_A_Z_0>
<VARIANCE> 39
33.543018 31.241779 36.076199 ...
```

```
<VARSCALE> 39
 2.974308e+01 4.143743e+01 3.819999e+01 ...
```

Perceptual Linear Prediction

- An alternative to the Mel-Frequency Cepstral Coefficients is the use of Perceptual Linear Prediction (PLP) coefficients
  - The mel filterbank coefficients are weighted by an equal-loudness curve
  - Compressed by taking the cubic root
  - From the resulting auditory spectrum LP coefficents are estimated
  - Converted to cepstral coefficients in the normal way

Energy Measures

- An energy term can be appended by including the qualifier _E in the target kind
- The energy is computed as the log of the signal energy

$$E = log \sum_{n=1}^{N} s_n^2$$

- This log energy measure can be normalised to the range $-E_{min}$ … 1.0 by setting the Boolean configuration parameter ENORMALISE to true
- The lowest energy in the utterance can be clamped using
- The configuration parameter SILFLOOR which gives the ratio between the maximum and minimum energies in the utterance in dB

Energy Measures

- When calculating energy for LPC-derived parameterisations, the default is to use the zeroth delay autocorrelation coefficient (r_o)
- This means that the energy is calculated after windowing and pre-emphasis
- If the configuration parameter RAWENERGY is set true, however, then energy is calculated separately before any windowing or pre-emphasis
- The qualifier o can be added to a target kind to indicate that the oth cepstral parameter C_o is to be appended

Delta, Acceleration and Third Differential Coefficients

- The qualifier _D indicates that first order regression coefficients (referred to as delta coefficients) are appended
- The qualifier _A indicates that second order regression coefficients (referred to as acceleration coefficients) are appended
- The qualifier _T indicates that third order regression coefficients (referred to as third differential coefficients) are appended
- The A qualifier cannot be used without also using the D qualifier. Similarly the T qualifier cannot be used without also using the D and A qualifiers

Delta, Acceleration and Third Differential Coefficients

- The delta coefficients are computed using the following regression formula

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2}$$

- The value of Θ is set using the configuration parameter DELTAWINDOW
- The same formula is applied to the delta coefficients to obtain acceleration coefficients
- At the beginning and end of the speech the default behavior is to replicate the first or last vector as needed to fill the regression window

Delta, Acceleration and Third Differential Coefficients

- For some purposes, it is useful to use simple differences throughout
- This can be achieved by setting the configuration variable SIMPLEDIFFS to true in HParm

$$d_t = \frac{(c_{t+\Theta} - c_{t-\Theta})}{2\Theta}$$

Storage of Parameter Files

- All parameterised speech data is stored externally in either native HTK format data files or Entropic Esignal format files

HTK format Parameter Files

- HTK format files consist of a contiguous sequence of samples preceded by a header
- The HTK file format header is 12 bytes long and contains the following data

nSamples	– number of samples in file (4-byte integer)
sampPeriod	– sample period in 100ns units (4-byte integer)
sampSize	– number of bytes per sample (2-byte integer)
parmKind	– a code indicating the sample kind (2-byte integer)

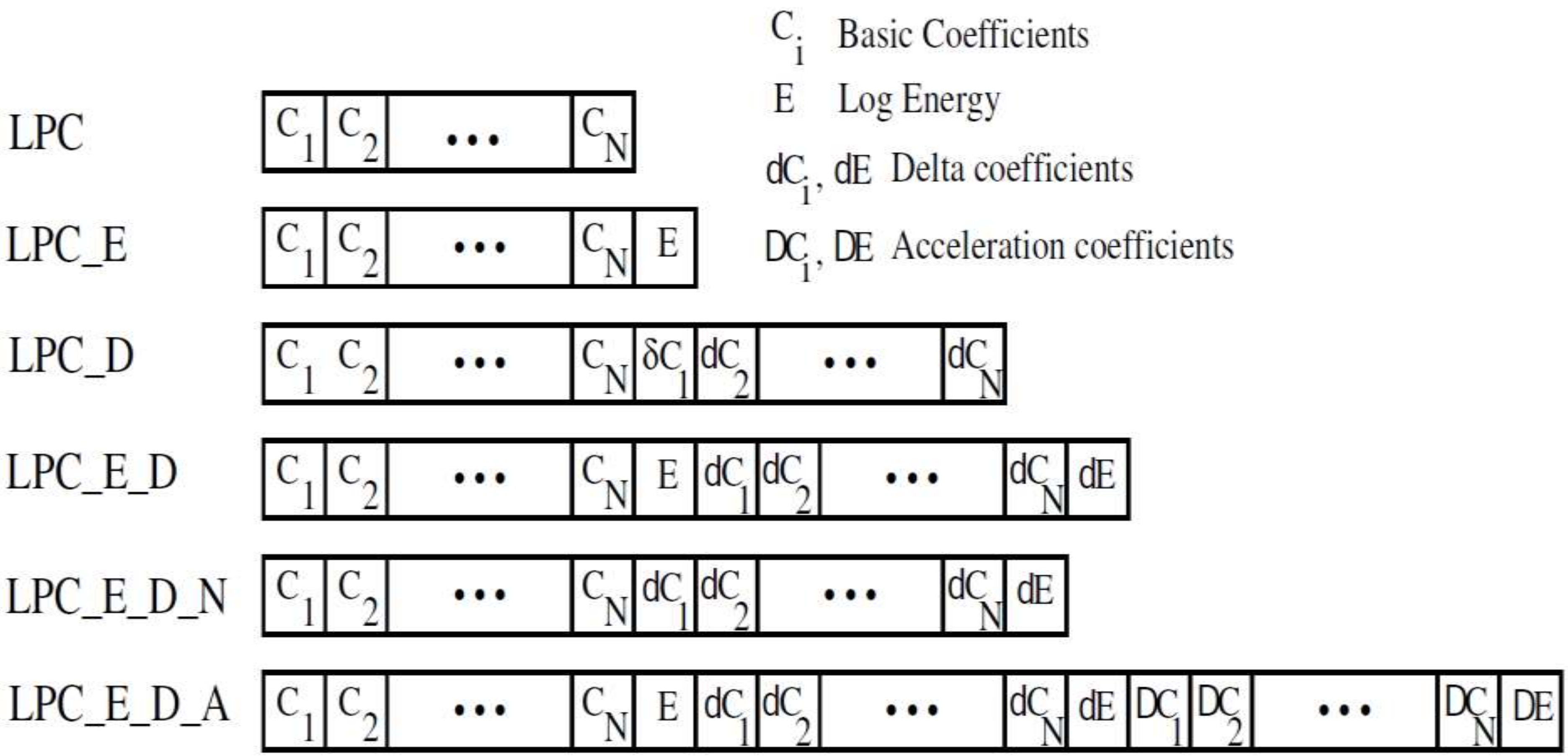
HTK format Parameter Files

0	WAVEFORM	sampled waveform
1	LPC	linear prediction filter coefficients
2	LPREFC	linear prediction reflection coefficients
3	LPCEPSTRA	LPC cepstral coefficients
4	LPDELCEP	LPC cepstra plus delta coefficients
5	IREFC	LPC reflection coef in 16 bit integer format
6	MFCC	mel-frequency cepstral coefficients
7	FBANK	log mel-filter bank channel outputs
8	MELSPEC	linear mel-filter bank channel outputs
9	USER	user defined sample kind
10	DISCRETE	vector quantised data
11	PLP	PLP cepstral coefficients

HTK format Parameter Files

_E	000100	has energy
_N	000200	absolute energy suppressed
_D	000400	has delta coefficients
_A	001000	has acceleration coefficients
_C	002000	is compressed
_Z	004000	has zero mean static coef.
_K	010000	has CRC checksum
_0	020000	has 0'th cepstral coef.
_V	040000	has VQ data
_T	100000	has third differential coef.

HTK format Parameter Files



HTK format Parameter Files

- For LP coding only, the IREFC parameter kind exploits the fact that the reflection coefficients are bounded by +/-1 and hence they can be stored as scaled integers such that +1.0 is stored as 32767
- For the general case

$$x_{short} = A * x_{float} - B$$

$$\begin{aligned} A &= 2 * I/(x_{max} - x_{min}) \\ B &= (x_{max} + x_{min}) * I/(x_{max} - x_{min}) \end{aligned}$$

Esignal Format Parameter Files

- ESIG files consist of three parts:
  - A preamble
  - A sequence of field specifications called the field list
  - A sequence of records
- The information in the preamble is the following

`line 1`	– identification of the file format
`line 2`	– version of the file format
`line 3`	– architecture (ASCII, EDR1, EDR2, machine name)
`line 4`	– preamble size (48 bytes)
`line 5`	– total header size
`line 6`	– record size

Esignal Format Parameter Files

- All ESIG files that are output by HTK programs contain the following global fields

commandLine the command-line used to generate the file;

recordFreq a double value that indicates the sample frequency in Herz;

startTime a double value that indicates a time at which the first sample is presumed to be starting;

parmKind a character string that indicates the full type of parameters in the file, e.g: `MFCC_E_D`.

source_1 if the input file was an ESIG file this field includes the header items in the input file.

- After that there are field specifiers for the records such as base kind of parameters, zeroc, energy, delta, delta zeroc, delta energy, accs, accs zeroc, accs energy
- The data segment have same format as HTK

Waveform File Formats

- HTK File Format/Esignal File Format
  - Same as the respective parameter file formats discussed before
- TIMIT File Format

`hdrSize`	– number of bytes in header ie 12 (2-byte integer)
`version`	– version number (2-byte integer)
`numChannels`	– number of channels (2-byte integer)
`sampRate`	– sample rate (2-byte integer)
`nSamples`	– number of samples in file (4-byte integer)

Waveform File Formats

- NIST File Format

sample_rate	– sample rate in Hz
sample_n_bytes	– number of bytes in each sample
sample_count	– number of samples in file
sample_byte_format	– byte order
sample_coding	– speech coding eg pcm, μlaw, shortpack
channels_interleaved	– for 2 channel data only

- The left/right channel only can be obtained by setting the environment variable STEREOMODE to LEFT/RIGHT

Waveform File Formats

- SCRIBE File Format
  - SCRIBE data files are headerless and therefore consist of just a sequence of 16 bit sample values
  - HTK assumes by default that the sample rate is 20kHz
- SDES1 File Format
  - The SDES1 header is complex (1336 bytes) since it allows for associated display window information to be stored in it as well as providing facilities for specifying repeat loops

Waveform File Formats

- SCRIBE File Format
  - SCRIBE data files are headerless and therefore consist of just a sequence of 16 bit sample values
  - HTK assumes by default that the sample rate is 20kHz
- SDES1 File Format
  - The SDES1 header is complex (1336 bytes) since it allows for associated display window information to be stored in it as well as providing facilities for

`headerSize`	– size of header ie 1336 (2 byte integer)
(182 byte filler)	
`fileSize`	– number of bytes of sampled data (4 byte integer)
(832 byte filler)	
`sampRate`	– sample rate in Hz (4 byte integer)
`sampPeriod`	– sample period in microseconds (4 byte integer)
`sampSize`	– number of bits per sample ie 16 (2 byte integer)

Waveform File Formats

- AIFF File Format
  - An AIFF file consists of a number of chunks
  - A Common chunk contains the fundamental parameters of the sound (sample rate, number of channels, etc)
  - A Sound Data chunk contains sampled audio data
- SUNAU8 File Format
  - Byte order must be set explicitly

`magicNumber`	– magic number 0x2e736e64
`dataLocation`	– offset to start of data
`dataSize`	– number of bytes of data
`dataFormat`	– data format code which is 1 for 8 bit μlaw
`sampRate`	– a sample rate code which is always 8012.821 Hz
`numChan`	– the number of channels
`info`	– arbitrary character string min length 4 bytes

Waveform File Formats

- OGI File Format
  - The OGI format is similar to TIMIT

`hdrSize`	– number of bytes in header
`version`	– version number (2-byte integer)
`numChannels`	– number of channels (2-byte integer)
`sampRate`	– sample rate (2-byte integer)
`nSamples`	– number of samples in file (4-byte integer)
`lendian`	– used to test for byte swapping (4-byte integer)

- WAV File Format

Waveform File Formats

'RIFF'	– RIFF file identification (4 bytes)
<length>	– length field (4 bytes)
'WAVE'	– WAVE chunk identification (4 bytes)
'fmt '	– format sub-chunk identification (4 bytes)
flength	– length of format sub-chunk (4 byte integer)
format	– format specifier (2 byte integer)
chans	– number of channels (2 byte integer)
sampsRate	– sample rate in Hz (4 byte integer)
bpsec	– bytes per second (4 byte integer)
bpsample	– bytes per sample (2 byte integer)
bpchan	– bits per channel (2 byte integer)
'data'	– data sub-chunk identification (4 bytes)
dlength	– length of data sub-chunk (4 byte integer)

Direct Audio Input/Output

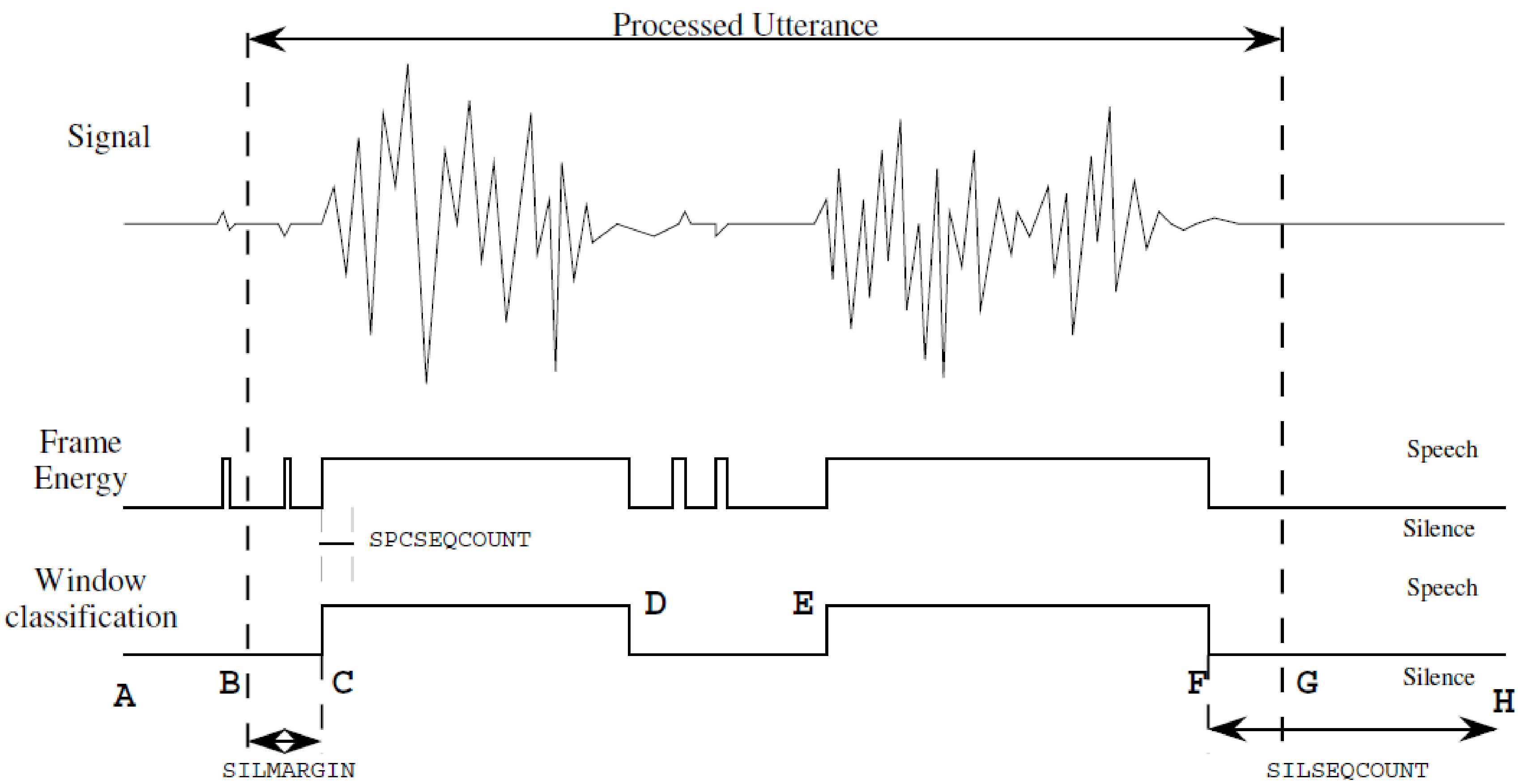
- Many HTK tools, particularly recognition tools, can input speech waveform data directly from an audio device
- The basic mechanism for doing this is to simply specify the SOURCEKIND as being HAUDIO
- HTK provides a number of Boolean configuration variables to request specific input and output sources

Variable	Source/Sink
LINEIN	line input
MICIN	microphone input
LINEOUT	line output
PHONESOUT	headphones output
SPEAKEROUT	speaker output

Direct Audio Input/Output

- The Haudio / Hparm modules provides two more powerful built-in facilities for audio input control
  - The first method involves the use of an automatic energy-based speech/silence detector which is enabled by setting the configuration parameter USESILDET to true
  - The second uses a signal to be sent from some other process

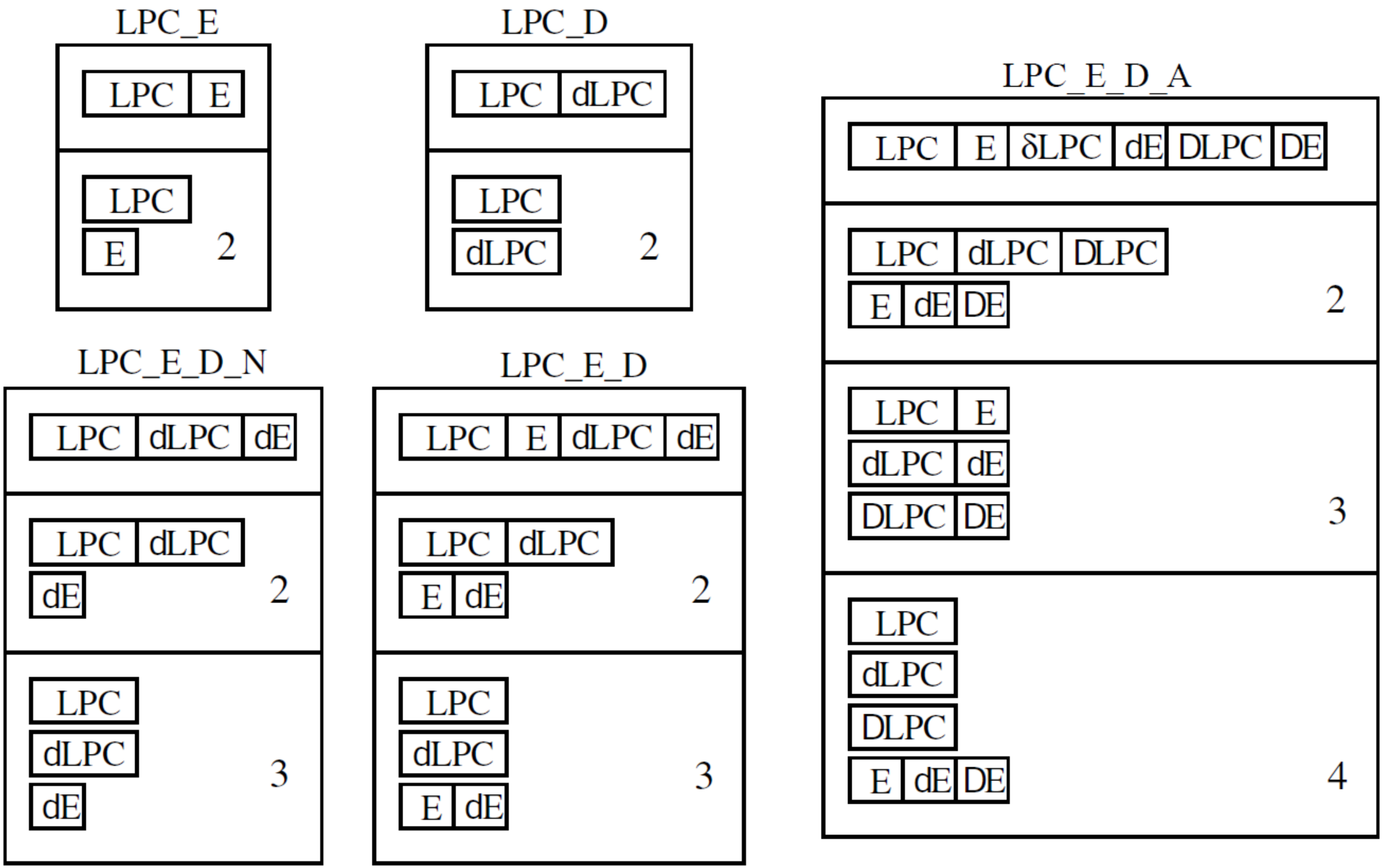
Direct Audio Input/Output



Multiple Input Streams

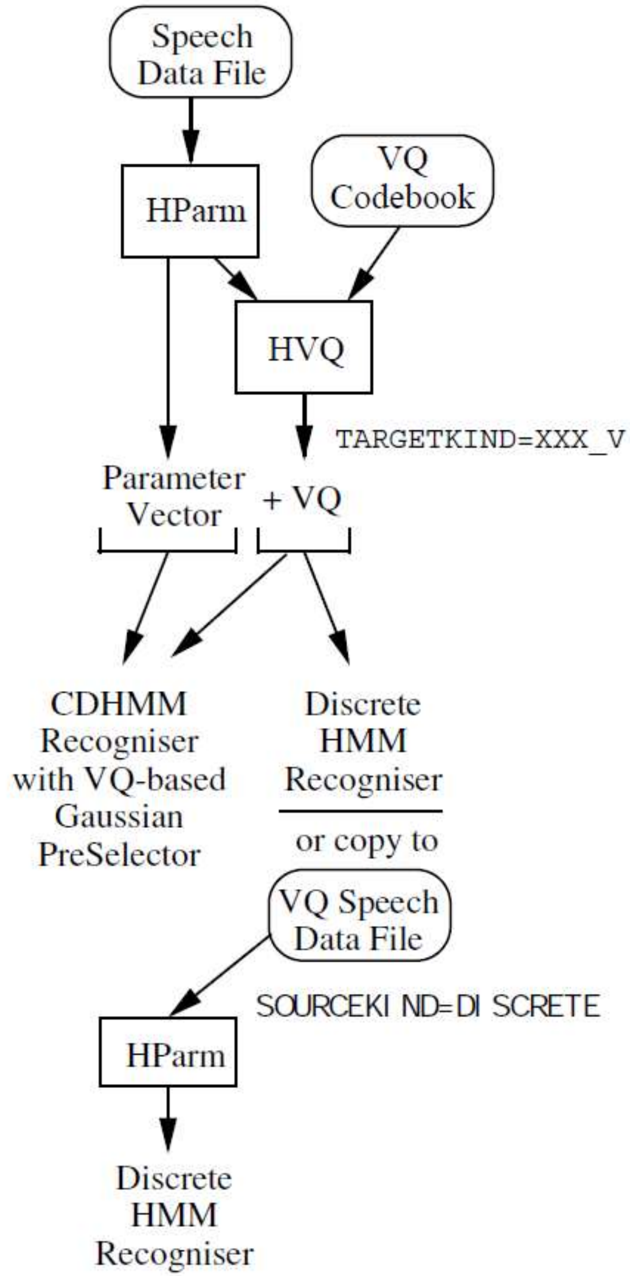
- HTK tools regard the input observation sequence as being divided into a number of independent data streams
- When building tied-mixture systems or when using vector quantization, a more uniform coverage of the acoustic space is obtained by separating energy, deltas, etc., into separate streams

Multiple Input Streams



Vector Quantization

- The HTK module HVQ provides a basic facility for performing this vector quantization
- The VQ table (or codebook) can be constructed using the HTK tool HQuant


Speech Data File
HParm
VQ Codebook
HVQ
TARGETKIND=XXX_V
Parameter Vector
+ VQ
CDHMM Recogniser with VQ-based Gaussian PreSelector
Discrete HMM Recogniser
or copy to
VQ Speech Data File
SOURCEKIND=DISCRETE
HParm
Discrete HMM Recogniser

Vector Quantization

- HVQ supports three types of distance metric
  - Euclidean
  - Full covariance Mahalanobis
  - Diagonal covariance Mahalanobis
- HVQ supports two organizations of VQ codebook
  - Linear
  - Binary Tree

Vector Quantization

Header

magic	–	a magic number usually the original parameter kind
type	–	0 = linear tree, 1 = binary tree
mode	–	1 = diagonal covariance Mahalanobis
		2 = full covariance Mahalanobis
		5 = Euclidean
numNodes	–	total number of nodes in the codebook
numS	–	number of independent data streams
sw1,sw2,...	–	width of each data stream

Nodes

stream	–	stream number for this node
vqidx	–	VQ index for this node (0 if non-terminal)
nodeId	–	integer id of this node
leftId	–	integer id of left daughter node
rightId	–	integer id of right daughter node
mean	–	mean vector
cov	–	diagonal variance or full covariance

Viewing Speech with HList

- Hlist can be used to display the contents of speech data files

```
HList -h -e 49 -F TIMIT timit.wav
----------------------------- Source: timit.wav -----------------------------
  Sample Bytes:  2          Sample Kind:    WAVEFORM
  Num Comps:     1          Sample Period:  62.5 us
  Num Samples:   31437      File Format:    TIMIT
------------------------------ Samples: 0->49 --------------------------------
    0:     8    -4    -1     0    -2    -1    -3    -2     0     0
   10:    -1     0    -1    -2    -1     1     0    -1    -2     1
   20:    -2     0     0     0     2     1    -2     2     1     0
   30:     1     0     0    -1     4     2     0    -1     4     0
   40:     2     2     1    -1    -1     1     1     2     1     1
------------------------------------ END ---------------------------------------
```

Viewing Speech with HList

- Hlist can be used to display the contents of speech data files

```
HList -s 5000 -e 5049 -F TIMIT timit.wav

---------------------------- Samples: 5000->5049 ----------------------------
 5000:     85   -116   -159   -252     23     99     69     92     79   -166
 5010:   -100   -123   -111     48    -19     15    111     41   -126   -304
 5020:   -189     91    162    255     80   -134   -174    -55     57    155
 5030:     90     -1     33    154     68   -149    -70     91    165    240
 5040:    297     50     13     72    187    189    193    244    198    128
------------------------------------ END ------------------------------------
```

Viewing Speech with HList

```
HList -C config -o -h -t -s 100 -e 104 -i 9  timit.wav
------------------------------ Source: timit.wav ---------------------------
  Sample Bytes:  2          Sample Kind:   WAVEFORM
  Num Comps:     1          Sample Period: 62.5 us
  Num Samples:   31437      File Format:   TIMIT
------------------------------------ Target --------------------------------
  Sample Bytes:  72         Sample Kind:   MFCC_E_D
  Num Comps:     18         Sample Period: 10000.0 us
  Num Samples:   195        File Format:   HTK
-------------------------- Observation Structure ---------------------------
x:     MFCC-1  MFCC-2  MFCC-3  MFCC-4  MFCC-5  MFCC-6  MFCC-7  MFCC-8       E
        Del-1   Del-2   Del-3   Del-4   Del-5   Del-6   Del-7   Del-8    DelE
------------------------------ Samples: 100->104 ---------------------------
100:    3.573 -19.729  -1.256  -6.646  -8.293 -15.601 -23.404  10.988   0.834
        3.161  -1.913   0.573  -0.069  -4.935   2.309  -5.336   2.460   0.080
101:    3.372 -16.278  -4.683  -3.600 -11.030  -8.481 -21.210  10.472   0.777
        0.608  -1.850  -0.903  -0.665  -2.603  -0.194  -2.331   2.180   0.069
102:    2.823 -15.624  -5.367  -4.450 -12.045 -15.939 -22.082  14.794   0.830
       -0.051   0.633  -0.881  -0.067  -1.281  -0.410   1.312   1.021   0.005
103:    3.752 -17.135  -5.656  -6.114 -12.336 -15.115 -17.091  11.640   0.825
       -0.002  -0.204   0.015  -0.525  -1.237  -1.039   1.515   1.007   0.015
104:    3.127 -16.135  -5.176  -5.727 -14.044 -14.333 -18.905  15.506   0.833
       -0.034  -0.247   0.103  -0.223  -1.575   0.513   1.507   0.754   0.006
------------------------------------- END ----------------------------------
```

Viewing Speech with HList

- Hlist can be used to display the contents of speech data files

```
HList -C config -n 3 -o -s 100 -e 101 -i 9  timit.wav
------------------------ Observation Structure -----------------------
nTotal=18 nStatic=8 nDel=16  eSep=T
x.1:    MFCC-1  MFCC-2  MFCC-3  MFCC-4  MFCC-5  MFCC-6  MFCC-7  MFCC-8
x.2:     Del-1   Del-2   Del-3   Del-4   Del-5   Del-6   Del-7   Del-8
x.3:         E    DelE
-------------------------- Samples: 100->101 -------------------------
100.1:   3.573 -19.729  -1.256  -6.646  -8.293 -15.601 -23.404  10.988
100.2:   3.161  -1.913   0.573  -0.069  -4.935   2.309  -5.336   2.460
100.3:   0.834   0.080
101.1:   3.372 -16.278  -4.683  -3.600 -11.030  -8.481 -21.210  10.472
101.2:   0.608  -1.850  -0.903  -0.665  -2.603  -0.194  -2.331   2.180
101.3:   0.777   0.069
---------------------------------- END ---------------------------------
```

Copying and Coding using HCopy

- HCopy is a general-purpose tool for copying and manipulating speech files

```
HCopy src tgt

HCopy src1 + src2 + src3 tgt

HCopy -s 100 -e -100 src tgt

HCopy -C config -s 100 -e -100 src.wav tgt.mfc
```

Script File

```
src1.wav tgt1.mfc
src2.wav tgt2.mfc
src3.wav tgt3.mfc
src4.wav tgt4.mfc
etc
HCopy -C config -s 100 -e -100 -S flist
```

Copying and Coding using HCopy

- HCopy is a general-purpose tool for copying and manipulating speech files

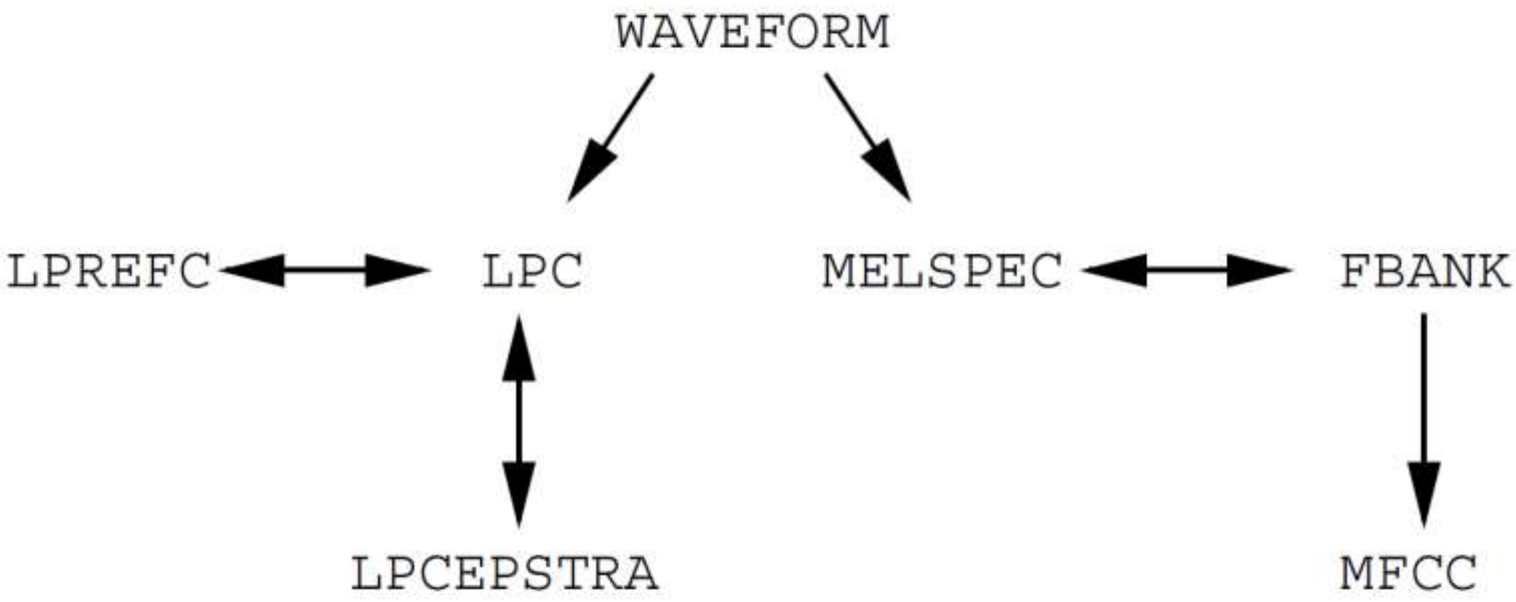
```
HCopy src tgt

HCopy src1 + src2 + src3 tgt

HCopy -s 100 -e -100 src tgt

HCopy -C config -s 100 -e -100 src.wav tgt.mfc
```

Script File

```
src1.wav tgt1.mfc
src2.wav tgt2.mfc
src3.wav tgt3.mfc
src4.wav tgt4.mfc
etc
HCopy -C config -s 100 -e -100 -S flist
```

Copying and Coding using HCopy

- HCopy is a general-purpose tool for copying and manipulating speech files



Thank You