Streaming Waveform Data Processing by Hermite Expansion for Text-Independent Speaker Indexing from Continuous Speech

Andrey S. Krylov, Danil N. Kortchagine, Alexey S. Lukin

Faculty of Computational Mathematics and Cybernetics, Moscow State University
Moscow, Russia

Abstract

In this paper we shall consider the new projection scheme of streaming waveform data processing for text-independent speaker indexing from continuous speech. It is based on an expansion into series of eigenfunctions of the Fourier transform. Partly this scheme can be also used for speech recognition.

Keywords: Fourier transform, Hermite functions, wave processing, speaker recognition.

1. INTRODUCTION

Fourier analysis plays a very important role in wave processing and wave analysis. At the same time, wave parameterization to code wave information by some kind of mathematical formulae enables to perform many of wave processing procedures in a most effective way. The aim of the work is streaming waveform data (Russian speech) processing by Hermite expansion for text-independent speaker indexing.

The proposed method is based on the features of Hermite functions and quasiperiods. An expansion of signal information into a series of these functions enables us to perform information analysis of the signal and its Fourier transform at the same time, because the Hermite functions are the eigenfunctions of Fourier transform. These functions are widely used in pure mathematics, where the expansion into Hermite functions is also called as Gram-Charlier series [1], [2] and image analysis [3-6]. It is also necessary to underline that the joint localization of Hermite functions in the both frequency and temporal spaces makes using this functions very stable to information errors. On the other hand, quasiperiod is a time period of the sound corresponding to period of base tone for vowels or resonant consonants, so extraction of quasiperiods suggests separating quasiperiods in a continuous speech waveform. So suggested method is very stable and flexible to using in streaming waveform data processing.

This work illustrates some possibilities to take full advantage of the use of this method.

2. HERMITE FUNCTIONS

The Hermite functions satisfy an important feature for wave processing, as they derivate a full orthonormal in $L_2(-\infty,\infty)$ system of functions.

The Hermite functions are defined as:

$$\psi_n(x) = \frac{(-1)^n e^{x^2/2}}{\sqrt{2^n n! \sqrt{\pi}}} \cdot \frac{d^n (e^{-x^2})}{dx^n}$$

They also can be determined by the following recurrent formulae:

$$\psi_0 = \frac{1}{\sqrt[4]{\pi}} \cdot e^{-x^2/2}$$

$$\psi_1 = \frac{\sqrt{2}x}{\sqrt[4]{\pi}} \cdot e^{-x^2/2}$$

$$\psi_n = x\sqrt{\frac{2}{n}} \cdot \psi_{n-1} - \sqrt{\frac{n-1}{n}} \cdot \psi_{n-2}, \forall n \ge 2$$

Moreover the Hermite functions are the eigenfunctions of the Fourier transform:

$$F(\psi_n) = i^n \psi_n,$$

where F denotes Fourier transform operator.

The graphs of the Hermite functions look like the following:

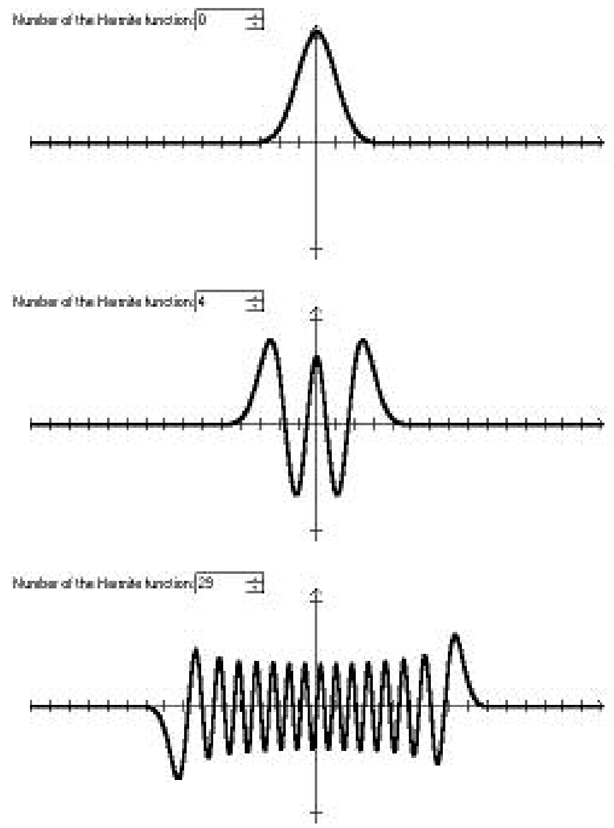

Figure 1: *Hermite functions*

3. HIERARCHY CODING IN HERMITE EXPANSION

The algorithm of hierarchy coding that we used looks as follows:

First, we approximate the whole quasiperiod with one function only, after that we subtract obtained result from original and repeat this operation with difference but using 2 functions. On every next step we use two times more functions than for the previous step (and stretch our interval of audio data to have all the Hermite functions used concentrated on the interval and to obtain the best approximation at this step). This channel separation with Hermite functions enables us to associate the obtained results with character formants.

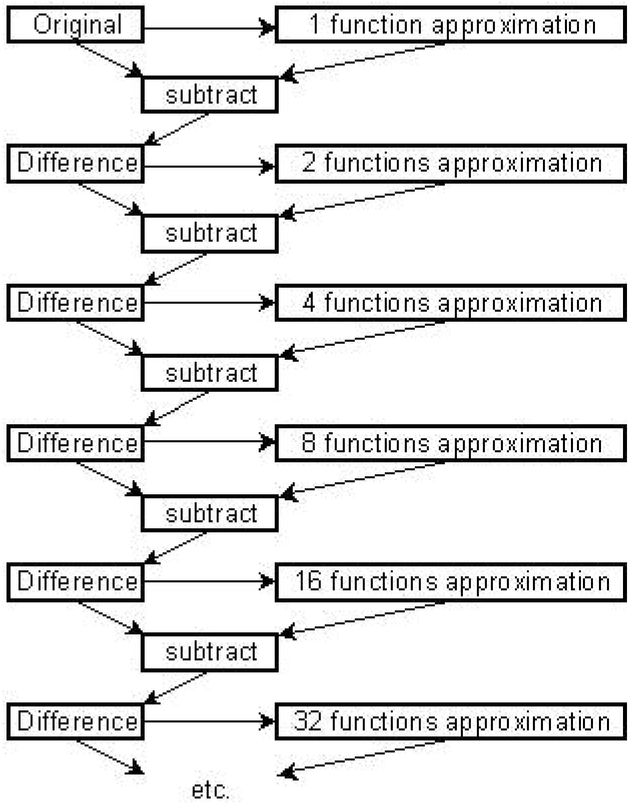
Original
1 function approximation
subtract
Difference
2 functions approximation
subtract
Difference
4 functions approximation
subtract
Difference
8 functions approximation
subtract
Difference
16 functions approximation
subtract
Difference
32 functions approximation
etc.

Figure 2: *Scheme of hierarchy coding*

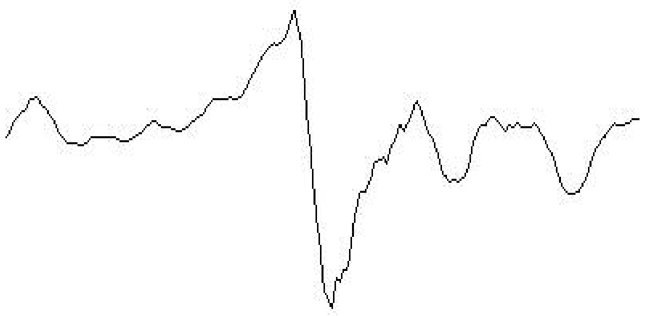

Figure 3: *Example of quasiperiod for character "o"*

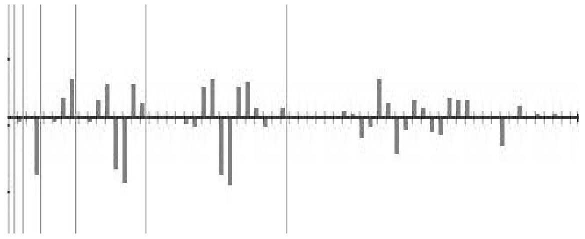

Figure 4: *Hermite expansion coefficients for quasiperiod above*

4. THE ALGORITHM

At this stage we have designed algorithm of text-independent speaker indexing with fixed/manual recognition threshold and the length of the time filter adjustment. Also vowels and resonant consonants indexing by this approach was designed to be used in speech recognition system. The default values of time filter length and recognition threshold used in this paper were 0.2 sec and 0.05 correspondently.

General scheme of the algorithm:

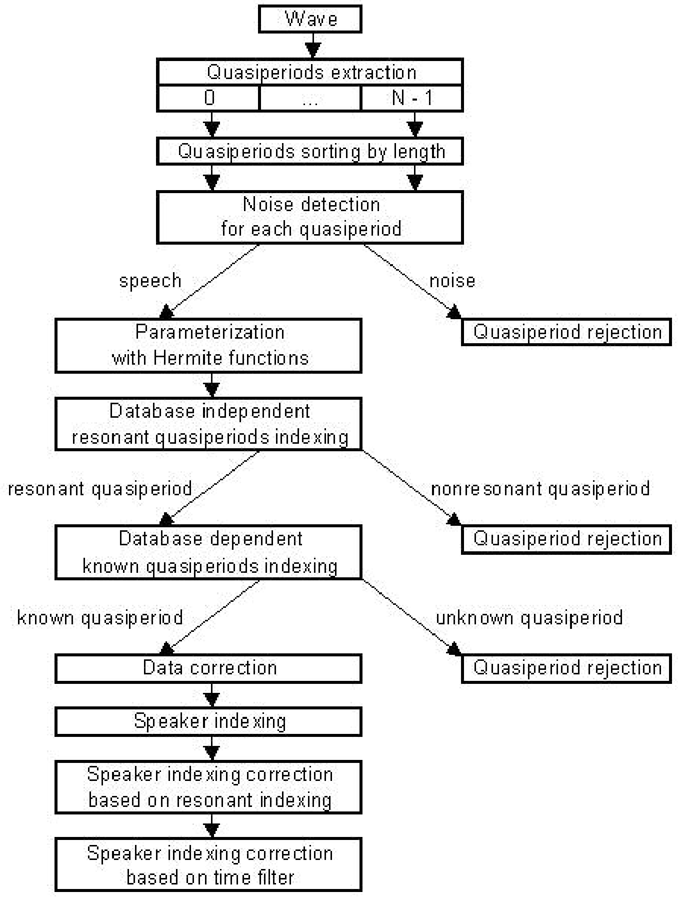
Wave
Quasiperiods extraction
0
...
N - 1
Quasiperiods sorting by length
Noise detection for each quasiperiod
speech
noise
Parameterization with Hermite functions
Quasiperiod rejection
Database independent resonant quasiperiods indexing
resonant quasiperiod
nonresonant quasiperiod
Database dependent known quasiperiods indexing
Quasiperiod rejection
known quasiperiod
unknown quasiperiod
Data correction
Quasiperiod rejection
Speaker indexing
Speaker indexing correction based on resonant indexing
Speaker indexing correction based on time filter

Figure 5: *Scheme of indexing algorithm*

At the first step we extract quasiperiods from the analyzed wave file and sort them by length (for improving performance).

After elimination of noise blocks (using noise threshold) we apply Hermite expansion for speech blocks (quasiperiods) at the second stage.

At the next step we index resonant quasiperiods using content-independent indexing algorithm.

Next we index resonant quasiperiods using database information and correct them depending on their arrangement.

The speaker indexing is performed using resonant quasiperiods. It is also based on using database information. The Hermite coefficients retrieved from database are compared with Hermite coefficients calculated for the given waveform.

Speaker indexing correction algorithms are used to treat incorrectly indexed speakers. First of these algorithms analyzes the correspondence between known resonant quasiperiods and indexed speakers to re-index segments with wrong detected speakers. Second algorithm is based on a use of time filter. Optimal time filter length for the dialogs used in this paper was found to be 0.2 sec.

This algorithm has been implemented in “Hermite coder PWE” software.

4.1 Quasiperiods extraction algorithm

4.1.1 Basic algorithm

Quasiperiod is a time period of the sound corresponding to period of base tone for vowels or resonant consonants. Extraction of quasiperiods suggests separating quasiperiods in a continuous speech waveform.

First of all the input waveform is pre-processed to increase algorithm robustness: DC (direct current) offset is eliminated, and (if needed) waveform can be inverted to make sharpest peaks point upside (see fig. 6). After that we scan the waveform to find a block with a largest RMS value (RMS window size = 13 ms). We suggest this block to be corresponding to a vowel, which has sharp quasiperiods (because sibilants and consonants usually have smaller RMS). Within this block we find a maximal positive sample value and consider it to be a starting boundary between two quasiperiods.

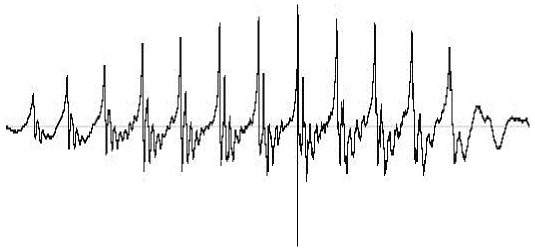

***Figure 6**: Now we have found the first boundary between quasiperiods. It will be a starting point.*

After that we continue finding quasiperiods’ boundaries at the rightmost part of the waveform. After the right part is processed, we reverse the left part of the waveform and find quasiperiods there using the same routine. The last step is to sort (simply rearrange) an array of quasiperiods’ boundaries.

Now we will describe how the main routine works. During each step we find the next border between quasiperiods (see fig. 7).

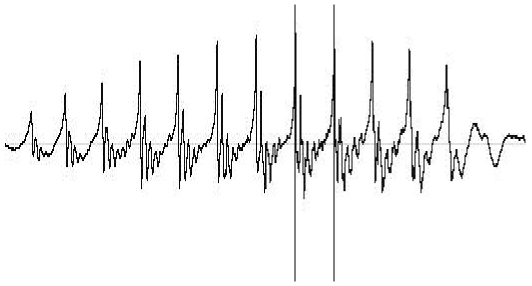

***Figure 7**: During each step we find the next border between quasiperiods.*

In our algorithm we find quasiperiods’ boundaries using a probability concept. For example if a number of previous quasiperiods were detected to have a base frequency around 120 Hz (1/T, where T is average duration), then we can expect that the next quasiperiod has the same duration because the fundamental frequency of human speech is varying relatively slow. If previous quasiperiods were detected with a high degree of confidence (see below), then the mentioned probability becomes even greater.

Given the current fundamental frequency $F_{current}$ we can suggest the most probable position for the next quasiperiod’s boundary. After that we construct a time-domain window, which has a shape of raised cosine, and apply this window to our waveform (see fig. 8), so that the center of the window was applied to the most probable boundary position.

$$WindowCenterOffset = \frac{1}{F_{current}}$$

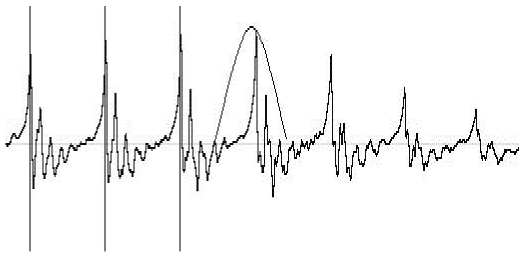

***Figure 8**: Raised cosine window increases the probability of preserving a frequency.*

Now we can find a maximum of the windowed waveform and decide it to be the next quasiperiod boundary. The current frequency is adjusted to match new conditions. Older quasiperiods also take part in the correction but now they have lower weights at the compound formula:

$$F_{current} = 0.7F_{current} + 0.3 \cdot \frac{1}{T_{last}}$$

Here T_{last} is the duration of the last detected quasiperiod.

After each 10 quasiperiods are found, the current frequency is adjusted by pitch detection routine (see later):

$$F_{current} = 0.5F_{current} + 0.5F_{detected}$$

If pitch detection routine fails to detect the pitch of the signal at the given period of waveform, then the current frequency is not changed.

The next step is to decide whether the last quasiperiod has been found with a high degree of confidence. To perform this task we check if there are any other maximums in the non-windowed (original) waveform during the last detected quasiperiod. If we can find a maximum that is higher than boundary values, then confidence degree is considered to be low, and the window selected for the next step will be wider than the current one. If the confidence degree is high (see fig. 7), we can choose a narrower window for the next step.

In this way we move throughout the input waveform till the end is reached.

4.1.2 Algorithm modification

Here we modify our algorithm so that quasiperiods' boundaries were corresponding not to maximums, but to zero-crossing points of waveform. This modification enables us to perform hierarchy coding of quasiperiods more effectively. We try to find a sample lying between two old boundaries, which is close enough to the center of the old quasiperiod and has a low absolute value. We use a V-shaped time window (see fig. 9) to increase a probability of snapping not only to zero, but also to the center of the old quasiperiod. After applying the window we search for a sample with absolute value, lower than a certain threshold (0.02 at our program). The search starts from the center of the widowed waveform to ensure that selected sample will lie as close as possible to the center.

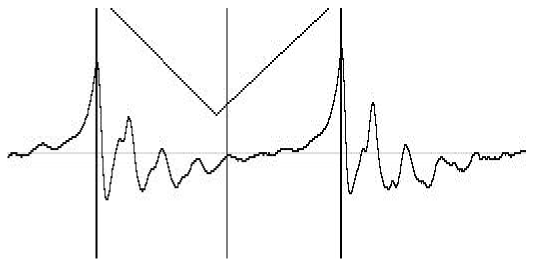

Figure 9*: Finding a new (zero-snapped) quasiperiods' boundary using a V-shaped window. Thick lines show old boundaries; thin line shows a new boundary.*

The center point of V-shaped window at the next old quasiperiod will be shifted to ensure preserving the frequency. An example of resulting quasiperiod boundaries by this algorithm is shown in fig. 10.

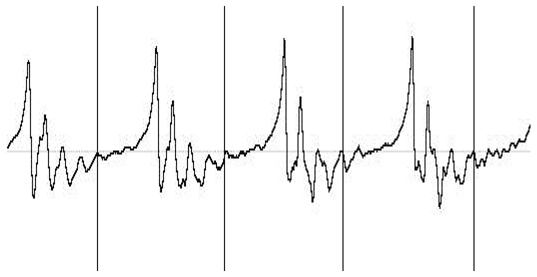

Figure 10*: Quasiperiods detected by a modified algorithm.*

4.1.3 Pitch detection algorithm

To decrease the error rate of quasiperiods' separation we compared two algorithms for estimating the pitch of speaker. Being able to extract the pitch of the signal locally, we can dynamically correct distances between quasiperiods' boundaries.

Two algorithms for pitch detection were considered. The first of them uses a spectrum of the signal to find the fundamental. The second one uses a cepstrum of the signal to analyze harmonic structure of the signal. The second algorithm has proved a better stability even when the fundamental is severely masked with overtones (like in a phone line). So it is currently used in our speaker indexing software and it is described below.

At first, the algorithm analyses the spectrum of the signal to decide whether the vocal or sibilant phoneme is present there. To decide the type of phoneme the algorithm calculates the averaged (per-frequency) energy at 2 frequency bands: from 200 to 2000 Hz and from 3800 to 10000 Hz. If the first energy is at least 8 times higher than the second one, we decide, that phoneme is vocal. If there is a sibilant phoneme, the fundamental cannot be detected.

The algorithm employs the cepstrum of the signal to analyze the periodic structure of the spectrum of the harmonic signal (see fig. 11). Periodic structure of spectrum corresponds to the fundamental frequency and harmonics with integer-multiple frequencies. When we take a Fourier transform of the logarithmic speech spectrum, we get a peak at the cepstrum, which corresponds to the fundamental frequency.

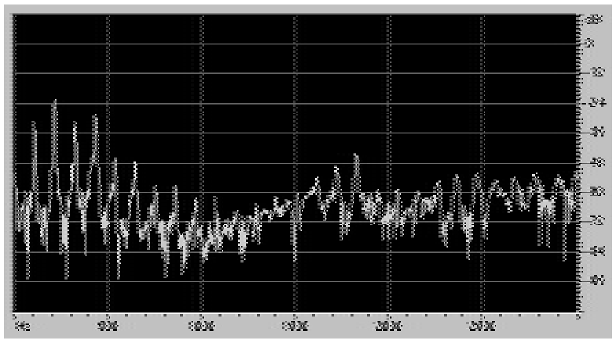

Figure 11: *Harmonic speech signal has a spectrum with a periodic structure.*

To achieve even better results, we use only real part of cepstrum (instead of a magnitude) for finding the peak, because the harmonic structure of spectrum corresponds to a zero-phase cosine harmonic.

This algorithm has shown stable results on a various speech material.

4.2 Speech/music/silence detection algorithm

An algorithm was developed for detecting silent or musical blocks at the file being processed. The algorithm helps us to exclude these blocks from speaker indexing process (see fig.12).

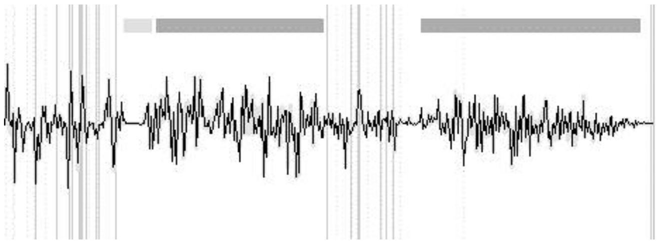

Figure 12: *The result of speech/music/silence detection algorithm. Detected music is marked with the dark horizontal bars; detected silence is marked with light horizontal bar. Vertical bars correspond to detected quasiperiods.*

At the first stage the algorithm detects silent blocks at the input waveform. Silent blocks usually have low level of signal and contain only background noise. To find such blocks more effectively we create a separate copy of input waveform which is preprocessed by high-pass FIR filter with a slow roll-off of 3 dB per octave. The filter transforms pink background noise into white noise and also cuts off low-frequency rumble. Then we get a waveform with significantly reduced amplitude of the noise and increased amplitude of sibilant consonants in speech. Without preprocessing sibilants are often lost in a low-frequency background noise.

After the filtering we apply an amplitude gate to the filtered waveform. The gate forces all the blocks with a volume below certain threshold to digital zero. The threshold is selected automatically. The gate features separate thresholds for opening and closing, and soft attacks and releases with look-ahead (attack) time of 125 ms and release time of 170 ms.

After the gating, the blocks with digital zeroes at the processed waveform are considered as silent blocks at the original waveform.

At the second stage the algorithm detects music at the input signal to exclude it from speaker indexing. The rest part of waveform is divided into 1-second intervals and fed to the algorithm of speech/music detection.

The decision is generated at each block separately. The final decision is a compound of 2 factors. The first factor is an amplitude dynamics of the block. The second factor is the distribution of fundamental in time at the block.

To calculate the first factor we obtain the amplitude envelope of the given block by calculating RMS with a moving 20 ms window. The standard deviations of RMS values over the entire block and over 3 sub-blocks (of 0.3 seconds) are measured. Their weighted sum is considered to be the first factor. Hereby we assume that speech usually has a wider dynamics than music has, because in speech we continuously observe fast changes of vowel phonemes with large amplitudes and short pauses between different phonemes or between words (see fig. 13). Music, in contrary, usually has low dynamics (esp. heavily compressed music on TV or radio).

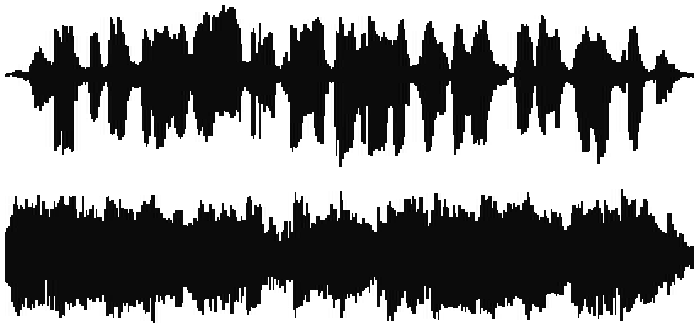
Figure 13: *Amplitude envelope of speech (upper) and music (lower). The dynamics of speech is significantly higher.*

To calculate the second factor for the given block we obtain the distribution of fundamental frequency in time. Here we use a slightly modified version of our pitch detection algorithm, featuring very high frequency accuracy. We obtain high degree of frequency accuracy by analyzing the harmonic structure of the signal. After we get the rough estimate of fundamental frequency from cepstrum, we analyze the spectrum of the signal to find higher harmonics. Because of the linear frequency grid, we get higher frequency resolution when analyzing higher harmonics. To find spectral peaks, corresponding to harmonics, even more precisely we use spline interpolation of spectrum values between FFT frequency bins. The final fundamental frequency is calculated as following:

$$F_0 = \frac{f_1 + f_2/2 + f_3/3}{3}$$

Here f_1, f_2 and f_3 are frequencies of 3 first harmonics (including the fundamental f_1) found at the interpolated spectrum.

After we calculate the fundamental frequency over the block with a step of 20 ms, we smooth the obtained array with a median filter (kernel size = 3) to eliminate dropped-off samples.

The next step is to decide, whether the obtained curve corresponds to speech or to music. To analyze the curve we introduce the dF_0 value, which is the first derivative of F_0 over time (actually we should write $\frac{dF_0}{dt}$). We calculate dF_0 value at each point of F_0 array and count the number of points, where the absolute value of dF_0 is between 30 and 500 Hz/second. Then at all such points we check the sign of dF_0 and exclude every point $dF_0(n)$, where

$$sign(dF_0(n)) \neq sign(dF_0(n-1))$$

or

$$sign(dF_0(n)) \neq sign(dF_0(n-2))$$

Then we calculate the percent of selected points and this percent is considered to be the second factor. Higher percent means that there's high probability that the given block corresponds to speech.

Here we assume that speech has a definite fundamental frequency, which is always varying in time with a speed from 30 to 500 Hz/second. The music, in contrary, usually has no definite fundamental frequency, so our routine for pitch detection either returns nothing, or resulting fundamental curve is not smooth (like for speech) and consists of random samples.

The final decision for each block is based on these 2 criterions (it is a weighted sum of them). The algorithm has shown good results on various sound samples. Still there are opportunities to improve its performance by carefully optimizing weights of different factors at the compound formula and performing more thorough spectrum analysis.

4.3 Vowels detection algorithm

A fast algorithm for detection of quasiperiods, which correspond to vowels in speech, has been developed. The algorithm works after the input waveform has been already separated into quasiperiods.

The final decision on each quasiperiod is based on 2 criterions: waveform shape similarity within 2 adjacent quasiperiods and a number of zero crossings at the given quasiperiod.

The first criterion estimates the shape similarity by finding the norm of the difference between waveforms of 2 adjacent quasiperiods.

$$Similarity = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}{\sum_{i=1}^{n} (x_i - y_i)^2}}$$

Here we assume that the shape of waveform corresponding to vowel is changing slowly (see fig. 10).

The second criterion calculates the number of zero-crossing points at the waveform of a given quasiperiod. If this number is between 2 and 15, then the probability of a vowel quasiperiod is high. Here we assume that sibilants and noise have a much larger number of zero-crossing points (because of significant HF component).

The final decision is calculated as a weighted sum of probabilities from these 2 criterions.

This algorithm has been integrated into our BSS program as a new criterion for speaker separation, and into our Hermite Coder program to exclude non-vocal quasiperiods from indexing and to increase its speed.

4.4 Approximated waves

At this stage, at first, we should select the number of the Hermite functions used for speaker indexing. The optimal number for this task using hierarchy coding is 63 functions (32 functions for the last layer). Further we stretch our approximation's quasiperiod [-A_0, A_0] to the segment [-A_1, A_1] for every layer, defined from the next criterion:

$$\int_{-A_1}^{A_1} \psi_n^{\,2}(x)dx = 0.99\,,$$

where n is the number of the Hermite functions per this layer for the approximation.

Then we decompose wave function *f(x)* into Fourier series by Hermite functions:

$$value_k(x) = \sum_{i=n_{k-1}}^{n_k - 1} c_i \psi_i(x)$$

$$c_i = \int_{-A_1}^{A_1} f_k(x)\psi_i(x)dx$$

$$n_k = 2^k, k = \overline{1,l},$$

where $f_k(x) = f_{k-1}(x) - value_{k-1}(x), f_0(x) = f(x),$

n_k is a number of functions for the current layer,

l is the current layer.

Since the Hermite functions are the eigenfunctions of Fourier transform, we have also found Fourier transform of the approximation for every quasiperiod of the original wave.

We used pre-sorting of the quasiperiods by length and odd/even property of Hermite functions to accelerate this algorithm.

4.5 Indexing results for three speakers conversation

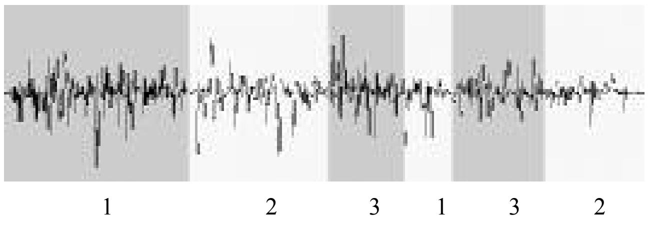

Figure 15: *Original dialog (15sec, 3 speakers, 5 changes)*

The shown waveform represents a conversation (in Russian) of 3 speakers on a news program on NTV Russian television. For every speaker we have trained a database on his independent monolog (length = 8 seconds). Each training took about 14 seconds (on PIII-750). Indexing took about 21 seconds. (It is necessary to notice, that even longer dialogues are calculated in real-time on PIII-850). Frequently we can reduce indexing errors by using manual threshold, but sometimes best threshold coincides with the default one. The example of reducing errors by using manual threshold you can see below (fig. 16, fig. 17).

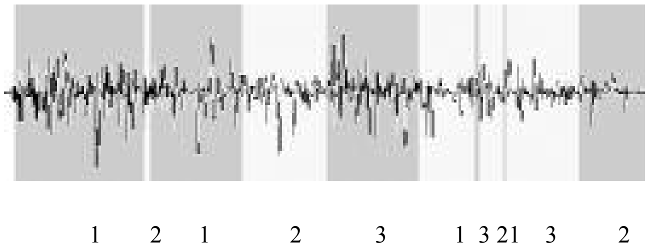

Figure 16: *Indexed dialog based on Hermite expansion (time filter is off, fixed recognition threshold = 0.05)*

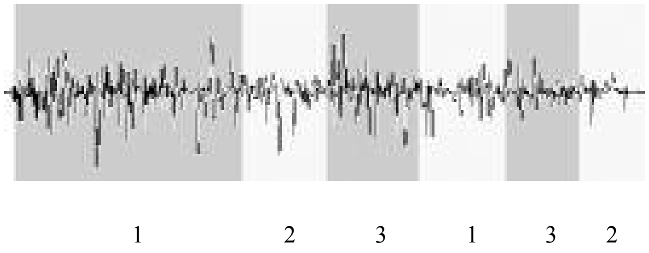

Figure 17: *Indexed dialog based on Hermite expansion (time filter is off, manual (best) recognition threshold = 0.0474)*

Let's compare results obtained by Hermite transform and results obtained by Fourier series (fig. 18, fig. 19).

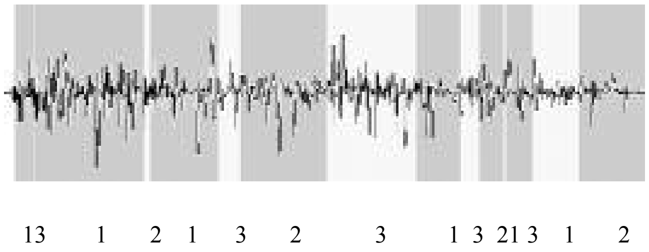

Figure 18: *Indexed dialog based on Fourier expansion (time filter is off, fixed recognition threshold = 0.05)*

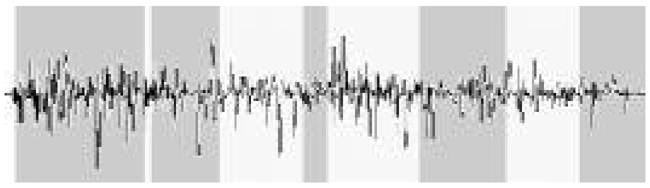

Figure 19: *Indexed dialog based on Fourier expansion*

(time filter is off, manual (best) recognition threshold = 0.042)

As we see, in this case application of Hermite transform is more justified, than application of Fourier series. As we will see further, statistics confirms this statement.

But this algorithm at this stage does not determine precisely indexing borders. It is due to processing only resonant consonants and vowels. Their distribution you can see on a figure 20.

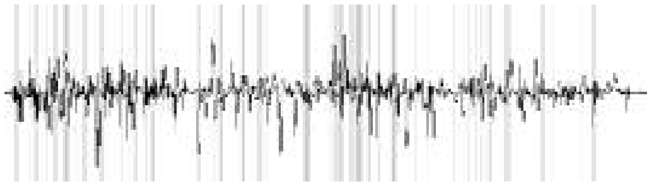

Figure 20: *Indexed resonant quasiperiods*

At figure 20 colored vertical bars correspond to sections of waveform consisting of resonant quasiperiods. The change of color indicates the change of a speaker or the change of phoneme.

At figures 15-19 colored vertical bars correspond to sections of waveform consisting of different speakers. The change of color indicates the change of a speaker. The numbers below figures show detected speakers.

As we saw before, thresholds are necessary to eliminate incorrect recognition of the periods lying between different phonemes. Minimal threshold is zero (all quasiperiods will be missed). Maximal threshold is one (every quasiperiod will be recognized). Optimal threshold found for the tested data is 0.05, but sometimes it must be corrected for better recognition.

Another way to eliminate incorrectly indexed speakers is to use time filter. Minimal time filter length is zero (time filter is off). Optimal found time filter length is 0.2 sec, but sometimes it must be corrected for better recognition. The example of reducing errors by using time filter for waveforms from figures 16 and 18 you can see on fig. 19 and fig. 20 correspondently.

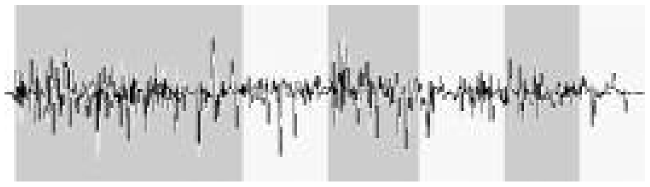

Figure 19: *Indexed dialog based on Hermite expansion*

(time filter length fixed to 0.2, fixed recognition threshold = 0.05)

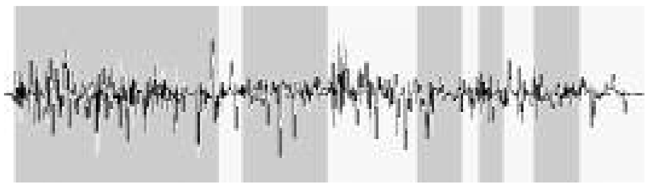

Figure 20: *Indexed dialog based on Fourier expansion*

(time filter length fixed to 0.2, fixed recognition threshold = 0.05)

As we see, in the case with time filter the application of Hermite transform is more justified too (compared to Fourier series). As we will see further, statistics confirms this statement too.

4.6 Statistics of indexing results

When acquiring statistics, the dialogues of two types were used: with two speakers and with three speakers. For each dialogue we also had a set of solo monologues of each of the speakers. The training monologues of two types were used: short monologues (7-15 seconds each), and long monologues (20-45 seconds each). All database trainings on these monologues were performed automatically. All dialogues were tested both using Hermite transform based indexing, and using Fourier series based indexing. In most cases it was necessary to correct the recognition threshold manually. In less often cases it was necessary to correct the length of a time filter manually.

We have calculated the error rate using the following formula:

$$er = \frac{\chi - \xi}{\chi}, \text{ where}$$

er is the error rate,

χ is the number of all inclusions of different speakers,

ξ is the number of correctly indexed inclusions of different speakers.

The obtained results are shown in the following tables:

Short training monologues (7-15 seconds each) without time filter:

Error rate	Fixed parameters	Manual parameters
Hermite transform	46,3%	**32,8%**
Fourier series	71,4%	52,3%

Short training monologues (7-15 seconds each) with time filter:

Error rate	Fixed parameters	Manual parameters
Hermite transform	43,2%	**7,6%**
Fourier series	61%	16,6%

Long training monologues (20-45 seconds each) without time filter:

Error rate	Fixed parameters	Manual parameters
Hermite transform	48,9%	**6,2%**
Fourier series	62,3%	8%

Long training monologues (20-45 seconds each) with time filter:

Error rate	Fixed parameters	Manual parameters

Hermite transform	10,5%	**4,4%**
Fourier series	31,5%	6,2%

Best achieved error rate without time filter is 6,2% (Hermite transform with long training monologues).

Best achieved error rate with time filter is 4,4% (Hermite transform with long training monologues).

It can be seen that the best error rate corresponds to Hermite transform with manual parameters adjustment. It is necessary to emphasize that quite good results are achieved at the default parameters on long trainings when time filter is turned on. Shown results confirm that when dynamically changing time windows are involved, the application of Hermite transform for speaker indexing is more justified.

5. CONCLUSION

In this paper we considered the new projection scheme of streaming waveform data processing for text-independent speaker indexing from continuous speech, which is based on the features of Hermite functions and quasiperiods. We have used an expansion into series of eigenfunctions of the Fourier transform, which has enabled us using advantages of a time-frequency analysis.

6. REFERENCES

[1] Gabor Szego "Orthogonal Polynomials". *American Mathematical Society Colloquium Publications,* vol. 23, NY, 1959.

[2] Dunham Jeckson, "Fourier Series and Orthogonal Polynomials". *Carus Mathematical Monographs*, No. 6, Chicago, 1941.

[3] Jean-Bernard Martens. "The Hermite Transform – Theory". *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 38 (1990) p. 1595-1606.

[4] Jean-Bernard Martens. "The Hermite Transform – Applications". *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 38 (1990) p. 1607-1618.

[5] Andrey Krylov and Danil Kortchagine "Projection filtering in image processing", *Graphicon'2000 Conference proceedings, Moscow* (2000) P.42-45.

[6] Andrey Krylov and Anton Liakishev. "Numerical Projection Method For Inverse Fourier Transform and its Application". *Numerical Functional Analysis and optimization*, vol. 21 (2000) p. 205-216.

About the authors

Dr. Andrey S. Krylov is the head scientist of Moscow State University.

E-mail: **kryl@cs.msu.su**

Danil N. Kortchagine is the student of Moscow State University.

E-mail: **dan_msu@euro.ru**

Alexey S. Lukin is the student of Moscow State University.

E-mail: **lukin@ixbt.com**

Address:

Faculty of Computational Mathematics & Cybernetics, Moscow State University, Vorob'evy Gory, 119992, Moscow, Russia.