On-Line Noise Suppression for Enhancing Speech Recognition

K. Urbanowicz, P. Fronczak and J.A. Hołyst
Faculty of Physics, Center of Excellence for Complex Systems Research, Warsaw University of Technology
Koszykowa 75, PL-00-662 Warsaw, Poland

(Received September 30, 2008)

An experimental setup has been designed that makes possible a real-time noise reduction and an increase of automatic word recognition rate. The applied algorithm uses a dynamical filtering method for investigated time-series and has been implemented in Spartan 3 FPGA evaluation board RC10 made by Celoxica. We show that our approach increases the number of correctly recognized words in the commercial speech recognition program ViaVoice 10.

PACS numbers: 05.45.Tp, 43.72.Dv, 43.50.Ki, 43.72.Ne

1. Introduction

Signals recorded by microphone are subjected to noise. There are several possible origins of this effect. The noise source can be due to the weak quality of the devices used to record (microphone, sound card, etc.). The second source of noise can be the environment surrounding the speaker, e.g. cars, working computers or other electronic devices. In the case of standard conditions of microphone, sound card and surrounding environment, the noise level can be about 40–50 dB (7–8 bits in a 16-bit recording). The character of the noise depends on the noise source. The microphone and the sound card produce rather high-frequency uncorrelated noise, but noise coming from the surroundings is usually more or less correlated. A method for removing the colored noise from a signal should depend also on the class of signal itself. If the signal's main frequency is much higher than the frequency of the colored noise, then the noise reduction (NR) is possible in some extent. However in the case of voice signal the spectrum of pitch frequency is very broad, so removing the colored noise is very difficult or even impossible. In fact, even a partial solution of this problem is demanding. In our approach we will try to remove high frequency uncorrelated noise with the help of a microprocessor that is operating in real time.

There are several methods of noise subtraction. Common solutions are linear filters; however one may also use nonlinear or hybrid filters. Among the linear filters the most popular is a low pass filter which is based on fast Fourier transform (FFT). An alternative for FFT is a simple time averaging method which also removes high frequency components but is not as precise as FFT in the Fourier space. In this paper we consider an algorithm belonging to the second class of noise suppression methods. We will show that our algorithm, described further as *dynamical filtering*, is able to denoise a recorded human voice signal so that the recognition rate of a commercial speech recognition program is enhanced. Hence, we use a corresponding quantifier for the rate of NR in human voice as it is described in Ref. [1]. We also verify that NR algorithms can help computers to enhance their intelligibility. This latter aspect is highly important, since every NR algorithm distorts the signal by removing noise. Hence enlarging the signal-to-noise ratio does not guarantee that a speech recognition program really understands the signal meaning better.

Nonlinear NR methods became a prominent issue about 15 years ago [2–12]. Since the analysis of chaotic data in terms of dimensions, entropies and Lyapunov exponents requires access to small length scales (small scale fluctuations of the signal), a moderate amount of measurement noise on data is already known to be destructive. On the other hand, a deterministic source of a signal, albeit potentially chaotic, supplies redundancy which enables one to distinguish between signal and noise and eventually to remove the latter to some extent. NR schemes which exploit such dynamical constraints were proposed in [2–12] and were tested for many data sets.

For our purposes more interesting are linear filters [13] which were discovered much earlier. The low pass filter based on Fourier transformation should suppress the power in a high frequency range in Fourier space in order to smooth the signal. A simpler case of such a filter is just averaging in space and the drawback is an additional artificial frequency which appears after the NR (the frequency is related to the number of average data).

In the next two sections we introduce basic definitions and compare features of different noise-reduction algorithms. Then we briefly recall our algorithm, including its adaptation for the treatment of voice data that are nonstationary limit cycle-like signals with embedded noisy segments (stemming from the fricatives). Then we demonstrate results from our algorithm including the analysis in Fourier space. At the end we present conclusions from our studies and suggest a possibility for future modifications.

2. Basic definitions

The level of noise is usually measured in dB. If a clean signal is $\tilde{x}(t)$, and an observed time series polluted by noise is $x(t)$, then the noise level is defined as the logarithm of the ratio of the signal power to the noise power, i.e.

$$N = 20\log_{10}\left(\frac{\langle x(t)^2\rangle}{\left\langle [x(t)-\tilde{x}(t)]^2\right\rangle}\right), \tag{1}$$

where $\langle . \rangle$ indicates the time average. Here we do not introduce a parameter of gain reduction because it would be rather meaningless and very difficult. The microprocessor changes the level of the signal amplitude so it could be not compared between the different types of reduction. In Table we recall the main features among different types of reduction schemes. For our purposes the most important are the second and third columns related to linear and dynamical filters, since both filters are compared in our studies. The nonlinear filters are much more complicated and at present it is not possible to implement them in the simple microprocessor that we tested in our experiments. We used Celoxica RC10 evaluation board equipped with a Xilinx XC2S1500 fpga, which is rather limited, and only less complicated formulae may be implemented.

TABLE

Comparison features of linear methods, dynamical filters, LP methods and hybrid methods.

Methods/Concepts	Linear filters (e.g. low pass filter)	Dynamical filters	Nonlinear methods (e.g. GHKSS)	Hybrid methods (e.g. LPNCF)
finding neighbours	in time	in time	in Fourier space	in time and in Fourier space
finding corrections	smoothing in time	smoothing in time changing according to the main frequency	smoothing in phase space	smoothing using the information in time and in space
noise estimation	in Fourier space	in Fourier space	violation of constraints in phase space	violation of constraints in time and in Fourier space

3. Methods used for speech signal processing

We present in this section some of the methods used for signal filtering. We need to stress, however, that the complexity of these methods is much too high for the microprocessor used in this study, so we cannot directly compare them to our approach tested with the microprocessor.

3.1. Linear filter

The most popular linear filter is the Wiener filter. It can be described as follows. If S_k^{signal} is the amplitude of Fourier transform of the noisy signal, then due to additivity and independence of the noise and signal we have

$$\left(S_k^{\text{signal}}\right)^2 = \left(S_k^{\text{noise}}\right) + \left(S_k^{\text{clean}}\right)^2. \tag{2}$$

The action of the optimal linear filter for white noise consists in rescaling the amplitudes of the signal in the Fourier space of the signal by the use of noise variance σ:

$$S_k^{\text{after}} = S_k^{\text{signal}}\left(\frac{\left(S_k^{\text{signal}}\right)^2 - \sigma^2}{\left(S_k^{\text{signal}}\right)^2}\right). \tag{3}$$

The inverse Fourier transform for the corrected amplitudes S_k^{after} keeps phases of the Fourier components of the noisy signal and yields the new signal in the time space. One can prove that such an algorithm is the optimal linear method of NR [13] if the exact value of noise level σ is known.

3.2. Nonlinear methods

The GHKSS method is a version of the local projective (LP) [1, 5–10, 12] method that was developed for chaotic signals corrupted by noise. Assuming that the clean data is confined to some deterministic attractor in a reconstructed state space, which itself is locally a subset of a smooth manifold, the method aims at identifying this local manifold in linear approximation and projecting the noisy state vector (which due to noise is not confined to this hyperplane) onto the local manifold. Algorithmically, this means identifying a neighbourhood in the delay embedding space and performing a singular value decomposition of a particular covariance matrix. Some refinements are described in [10].

The LPNCF method, which was particularly developed for chaotic flows, makes use of nonlinear constraints which appear because of the time continuous character of the flow. These constraints are functionals of the state vectors, which vanish for dense sampling of deterministic data. Let $\{x_i\}$ for $i = 1, 2, \ldots, N$ be the time series. We denote the corresponding clean signal by $\{\tilde{x}_i\}$, so for the measurement noise $\{\eta_i\}$ we have $x_i = \tilde{x}_i + \eta_i$ where $i = 1, 2, \ldots, N$. We define the time delay vectors $x_i = (x_i, x_{i-1}, \ldots, x_{i-(d-1)})$ as our points in the reconstructed phase space. Then we choose two neighbours $x_k, x_j \in \mathcal{X}_n^{\mathrm{NN}}$ of the vector x_n ($\mathcal{X}_n^{\mathrm{NN}}$ is the set neighbours of the point x_n). Let us introduce the following function [12]:

$$\begin{aligned} G_n(s) &= x_{n-s}(x_{k+1-s} - x_{j+1-s}) \\ &\quad + x_{k-s}(x_{j+1-s} - x_{n+1-s}) \\ &\quad + x_{j-s}(x_{n+1-s} - x_{k+1-s}) \end{aligned} \tag{4}$$

for $s = 0, 1, \ldots, d-1$.

The function $G_n(s)$ vanishes for clean one-dimensional systems, since we can eliminate auxiliary parameters a and b from appearing in the following equations:

$$\tilde{x}_{k+1} = a\tilde{x}_k + b. \tag{5}$$

In the case of higher dimensional systems the function $G_n(s)$ does not always vanish, but it does alter slowly in time for dense sampling.

One can check that for a highly sampled clean dynamic, the following constraint can be introduced:

$$\mathcal{C}_n^q = \sum_{k=0}^{q-1} (-1)^2 G_n(k) \approx 0$$

$$l = \begin{cases} 0 & \text{if} \quad k = 0, \\ k + \sum_{s=1}^{\mathrm{int}(\log_2(k))} \mathrm{int}\left(\frac{k}{2^s}\right) & \text{if} \quad k > 0, \end{cases} \tag{6}$$

where $\mathrm{int}(z)$ is an integer part of z and $\log_2(z)$ is a logarithm with a base 2 of z. Similarly to LP methods, the constraints (6) are satisfied in this approach by application of the method of Lagrange multipliers to an appropriate cost function. Since we expect that corrections to noisy data should be as small as possible, the cost function is assumed to be the sum of squared corrections $S = \sum_{s=1}^{N} (\delta x_s)^2$, where δx_n are the corrections of the NR method connected to x_n, such that the resulting time series of the NR method y_n is defined as $y_n = x_n + \delta x_n$ ($n = 1, 2, \ldots, N$). The method is a compromise between time and space integration methods. The introduced constraints include neighbours in space together with their pre-images, and it works on a time lag of unity in the embedding space in order to exploit the flow-like structure of the data. Hence it combines spatial and temporal vicinity (see Table). It can perform better than standard time averaging or standard LP methods, because the size of the neighbourhood in time and in space is smaller in the LPNCF method than in standard methods which use only time or space averaging.

4. Description of the algorithm

It is known that the voice signal for most of time has many similarities to a flow [1, 14, 15], which means that it represents smooth anharmonic oscillations with a typical frequency around the speaker's pitch of a few hundreds Hz. However, articulated human voice is a concatenation of different phonemes, so that the frequency, amplitude and, most importantly, the wave form of the oscillation various tremendously on time scales of about 50 to 200 ms, causing the signal to be highly nonstationary.

Our method is based on a simple averaging process, but in contrast to low pass filter the average scale depends on the pitch frequency of the signal itself. The selected averaging period is simply 1 over 120 times pitch frequency, while the pitch frequency is calculated using signal covariances. Human speech appears as a result of periodic behavior of parts of the signal, which is repeated a few tens of times. One can recognize the main frequency that is the pitch frequency. A rapid change of the oscillation period usually means the end of the vowel, consonant or syllable. With the help of our algorithm we try to catch this dominant frequency and set the averaging period.

The recognition program is based on the Fourier transform. It tries to find signals in its own databank scheme in the Fourier space for the closest word to the current signal. The recognition program also takes into account the high frequency domain. This is more evident especially for signals from noisy letters like "s", "th", etc. The NR algorithm has to be very careful with the chosen level of averaging and should not destroy the high frequency voice signal. To have a better result, we tried to reduce every syllable separately. We found the optimal number of points in the averaging process to be $T/120$, where T is the inverse of pitch frequency.

The programming of the microprocessor is different from high level software, as C or C++. The most important difference is that the microprocessor uses only integers, while floating numbers are forbidden. The second important difference is that the microprocessor can perform parallel tasks. The third point is that the programmer should use bit operations on numbers rather than standard operations and functions. These restrictions make the programming of the microprocessor much harder, and developing even a simple code can be difficult.

In Fig. 1 we show the global scheme of the microprocessor computations. The scheme consists of four parts which work in parallel. The microprocessor takes data, calculates the averaging period, averages and writes the reduced signal to headphones entry, all at the same time. All tasks are synchronized with signal sampling (22 kHz). Here the averaging means a simple arithmetic mean value for a given period. The important part of the calculations is hidden in the second square — the estimation of pitch

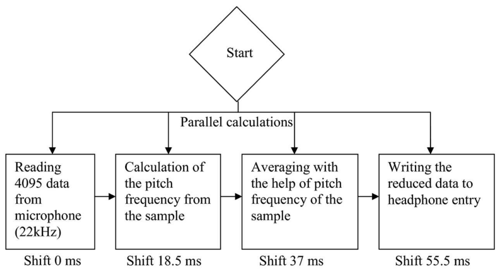
Start
Parallel calculations
Reading 4095 data from microphone (22kHz)
Calculation of the pitch frequency from the sample
Averaging with the help of pitch frequency of the sample
Writing the reduced data to headphone entry
Shift 0 ms
Shift 18.5 ms
Shift 37 ms
Shift 55.5 ms

Fig. 1. Scheme of parallel computing in the microprocessor.

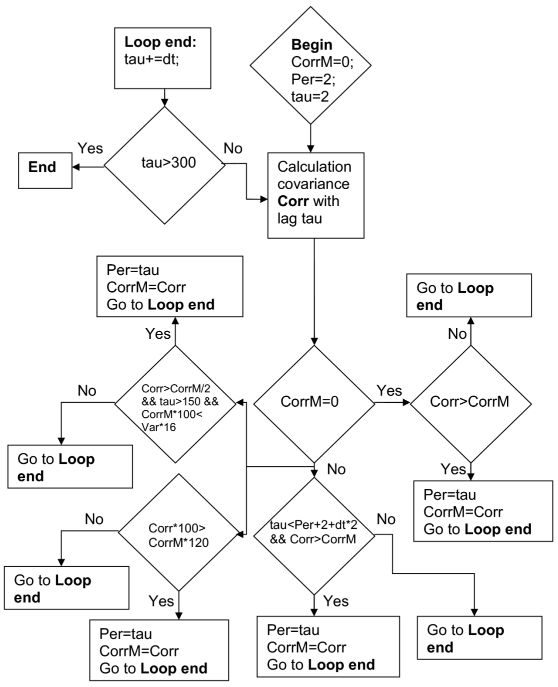
Loop end: tau+=dt;
Begin CorrM=0; Per=2; tau=2
tau>300
Yes
No
End
Calculation covariance Corr with lag tau
Per=tau CorrM=Corr Go to Loop end
Go to Loop end
Yes
No
No
Corr>CorrM/2 && tau>150 && CorrM*100< Var*16
CorrM=0
Yes
Corr>CorrM
Go to Loop end
No
Yes
Per=tau CorrM=Corr Go to Loop end
No
Corr*100> CorrM*120
tau<Per+2+dt*2 && Corr>CorrM
No
Go to Loop end
Yes
Yes
Per=tau CorrM=Corr Go to Loop end
Per=tau CorrM=Corr Go to Loop end
Go to Loop end

Fig. 2. Calculations of the pitch frequency of the sample with the help of covariance.

frequency. We present the method of finding the frequency in Fig. 2. The frequency is simply the inverse of the time span of the unchanged pitch (the time between repetitions). As we see in the scheme, there are many constant parameters which come out of the experimental optimization for characteristic features of the human voice. Here we recall once more that for microprocessors one can use only integer numbers. The frequency finding algorithm is done in a loop. This loop is within a higher level loop, which is based on frequency and tries to catch the end of the syllable in the rapid change of this frequency. The outer loop is shown in Fig. 3. Looking closely at Fig. 3 we see that if the parameter PerF (an average of Per) is changing rapidly, then we break the loop and return to the end of the syllable and frequency. Calculations of covariance and variance are presented in Eqs. (7) and (8):

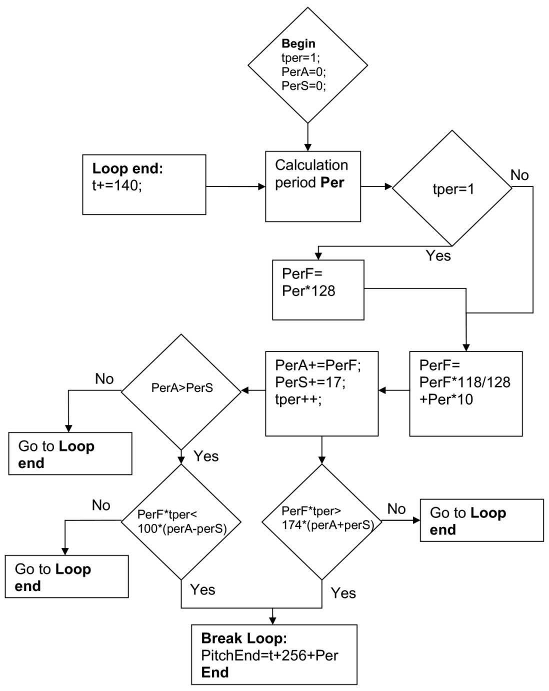
Begin tper=1; PerA=0; PerS=0;
Loop end: t+=140;
Calculation period Per
tper=1
No
Yes
PerF= Per*128
PerF= PerF*118/128 +Per*10
PerA+=PerF; PerS+=17; tper++;
PerA>PerS
No
Go to Loop end
Yes
No
PerF*tper< 100*(perA-perS)
PerF*tper> 174*(perA+perS)
No
Go to Loop end
Go to Loop end
Yes
Yes
Break Loop: PitchEnd=t+256+Per End

Fig. 3. Calculation of the beginning and end of a syllable.

$$\mathrm{Corr} = \frac{1}{256}\sum_{i=1,2,\ldots}^{256} x_i x_{i+\mathrm{tau}}, \tag{7}$$

$$\mathrm{Var} = \frac{1}{256}\sum_{i=1,5,\ldots}^{1024} x_i x_i, \tag{8}$$

where x_i is the value of voice amplitude at time i.

One more important point in Fig. 2 is the problem with silence (pause) periods, which is solved as follows. The condition if ($\mathrm{Corr} > \mathrm{Corr\,M}/2$ && $\mathrm{tau} > 150$ && $\mathrm{Corr\,M} * 100 < \mathrm{Var} * 16$) detects silence periods. If the algorithm recognizes no voice in the signal, then it sets the frequency to the lowest value, so the period of reduction will be maximal and the whole noise will be removed. This seems to be an important feature of the method, because the recognition algorithm is frequently misled by the noise in the pause periods. The method was described in detail in the paper [1], and here we introduced the changes needed to adopt the algorithm to microprocessor properties.

5. Results

The results were obtained with the experimental setup shown in Fig. 4. The voice was played back from files so one could repeat the recognition procedure for different parameters of reduction. We performed the recognition for the following cases: no noise reduction, reduction by a standard method (low pass filter) and reduction by the method described above — dynamic filtering. We repeated the recognition four times for every case in order

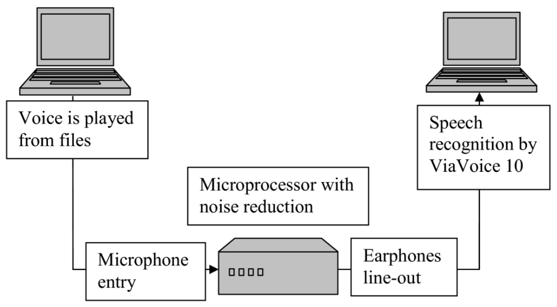
Voice is played from files
Speech recognition by ViaVoice 10
Microprocessor with noise reduction
Microphone entry
Earphones line-out

Fig. 4. Scheme of experiment setup of reduction of signal with voice.

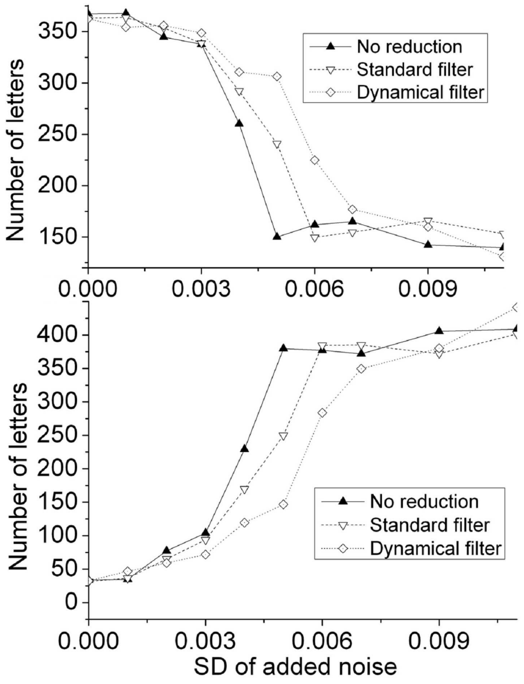

Fig. 5. Plots of intelligibility. At the Y-axis, the number of letters similar to (top) and different from (bottom) the original clean text versus level of distortion is shown. SD stands for standard deviation. The graphs shift to the right when a proper reduction is applied, which means that intelligibility is enhanced.

to average the results. The standard method is equivalent to simple time averaging. Here we performed averaging with three points which we found optimal for this type of reduction and it is the average number of data taken for our method. As mentioned above, our method reduces the noise dynamically, which means that it fits dynamically the number of data for signal averaging. In the case of our experimental setup the average ranges from 1 to 7 and it depends on characteristic frequency, which is detected by the method. The number of taken data to average suggests that we are dealing with high frequency noise which can be partially reduced by averaging of a few data. Figure 5 presents the main results in

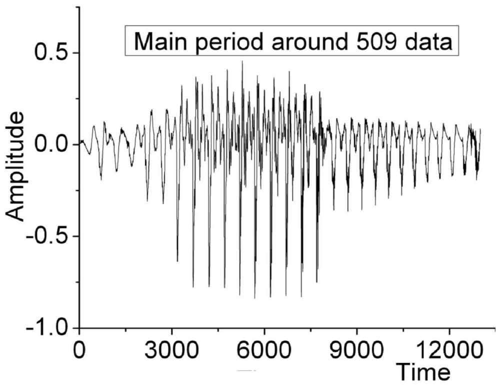

Fig. 6. The time series corresponding to the low frequency signal of the beginning of word "München".

the case of recognition. The intelligibility of the signal is estimated using two values: the number of correctly recognized letters and the number differences between the cleaned signal and the original text, which is known a priori. As one can see the signal converted by the microprocessor but not cleaned leads to the worst results: there are the largest differences and the smallest similarities between the system output and the original text. When we use a standard method of NR (low pass filter) the intelligibility is better than for the uncleaned signal and the observed shift of plot to the right means that the signal is recognized for larger noise. In our test we used a clean signal with standard deviation 0.02355. In the case of no NR the recognition program is completely misled, for a noise standard deviation SD = 0.005 (26.92 dB), while for the simple reduction the threshold is shifted to SD = 0.006 (23.75 dB). Our method of NR makes the signal even more readable to the computer, and the noise threshold is shifted to SD = 0.007 (21.07 dB). This means that the computer can recognize the signal if the noise standard deviation is smaller than 0.007. The recognition program is fully misled for higher values of noise, which is represented on the plots by the horizontal line, without substantial changes of recognition for increasing noise intensity. Let us stress that Figs. 5 and 6 cannot be generated by simple inverting, since the speech recognition program ViaVoice [16] can add additional words to the text, for example when the end of a line should occur. This is due to the fact that this program recovers whole words from the signal, so number of letters is not preserved and it is rather increased when the voice distortion is larger.

In order to look deeper into properties of considered time series we calculated and compared power spectra of different types of signal for various reduction methods. Since the speech recognition program is based on FFT calculations and the comparison in Fourier space is more appropriate for this case, we took two sample signals with extreme values of main frequency. One signal possesses a main period length equal to 509 data points for the case of 32 kHz sampling (low frequency signal,

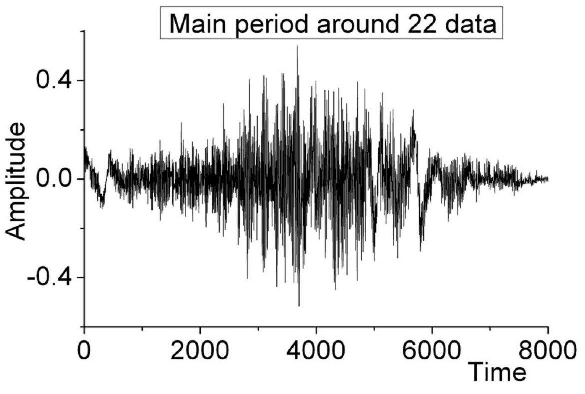

Fig. 7. The time series corresponding to high frequency signal of the noisy letter “ch” in the word “München”.

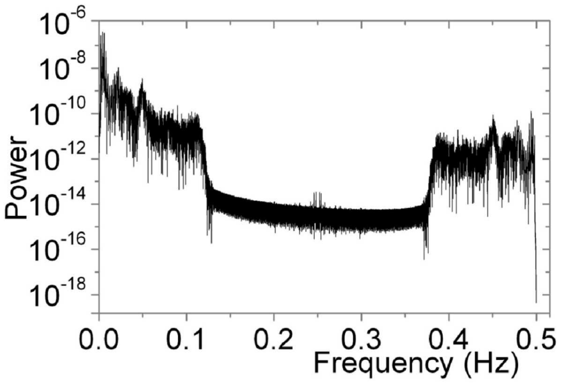

Fig. 10. The power spectrum of noisy low frequency signal without reduction.

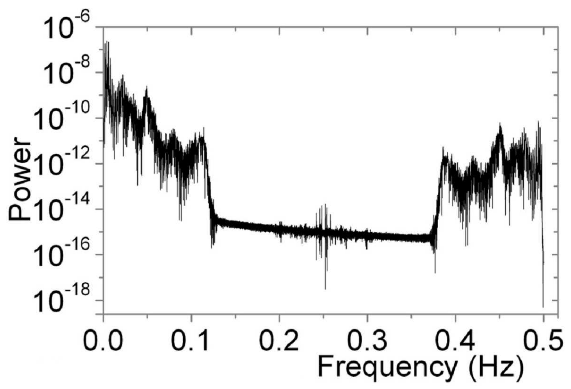

Fig. 8. The power spectrum of clean low frequency signal.

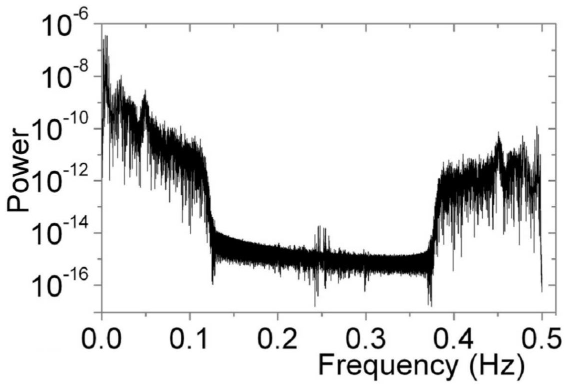

Fig. 11. The power spectrum of noisy low frequency signal with standard filtering.

$f = 62$ Hz) and it corresponds to the beginning of the word “München” (see Fig. 6). The second sample has a main period length equal to 22 data points (high frequency signal, $f = 1.45$ kHz) and it is the noisy letter “ch” in the same word (see Fig. 7). We applied two types of NR to the above signals: standard filtering and dynamic.

We also considered the signal with no NR (we used a noise with a standard deviation SD = 0.005) and a clean signal. In Figs. 8 and 9 we present the power spectrum of clean signal for low and high frequency data, respectively. The difference between these figures is understandable — more power in high frequencies is present for the second

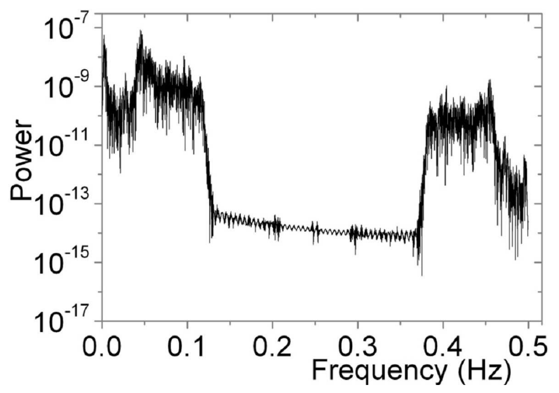

Fig. 9. The power spectrum of clean high frequency signal.

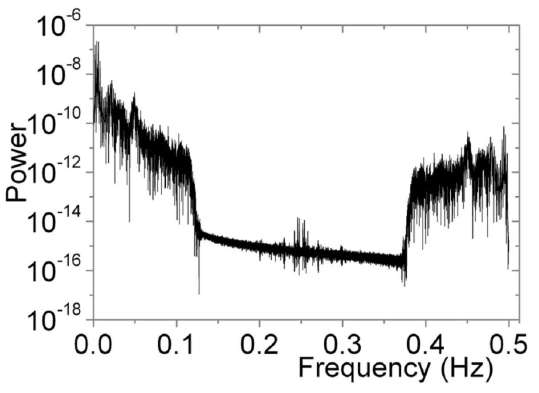

Fig. 12. The power spectrum of noisy low frequency signal with dynamic filtering.

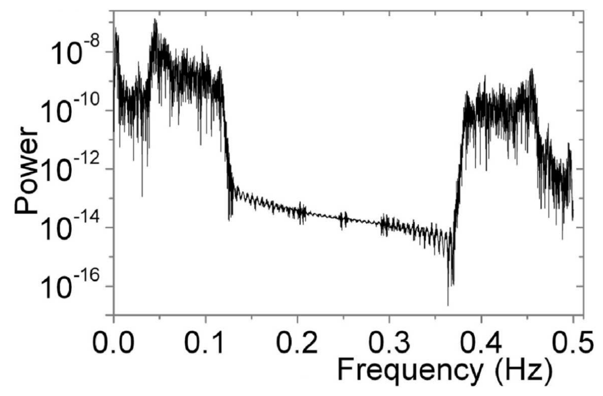

Fig. 13. The power spectrum of noisy high frequency signal without reduction.

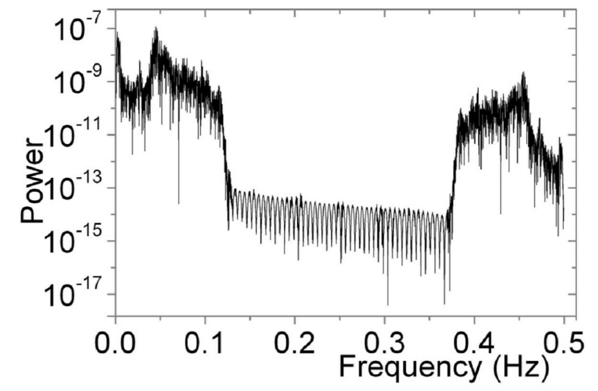

Fig. 14. The power spectrum of noisy high frequency signal after standard filtering.

picture. Figures 10, 11 and 12 compare effects of the standard filtering and the dynamic one for the case of low frequency signal. For recognition purposes we would like to smooth the signal much more than for high frequency. The comparison from the beginning of the power spectrum till 0.13 brings more similarities to Fig. 11 and Fig. 12, so dynamic filtering behaves like standard filtering and substantially reduces the noise. The shape

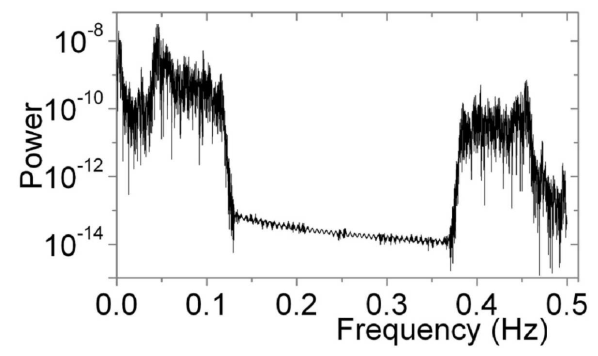

Fig. 15. The power spectrum of noisy high frequency signal after dynamic filtering.

of the power spectrum is smoother and the details vanish. In Figs. 13, 14, and 15 we compare the effects of standard filtering and dynamic filtering for the case of high frequency signal. In this case the power spectrum of the signal after the dynamic filtering is more similar to the case of the signal without NR (Fig. 15 is similar to Fig. 13). Details in the power spectrum are preserved so the speech recognition program is not substantially misled. It follows that the dynamic filtering changes the level of averaging in order to preserve important details in the power spectrum for the speech recognition algorithm.

6. Conclusions and further discussion

In this paper we describe an experiment in NR using a method of dynamic filtering and Spartan 3 FPGA evaluation board RC10 made by Celoxica. The experiment showed that the proposed NR method enhances the possibility of word recognition and amplifies the possibility of communication with computers. A commercial program for speech recognition, ViaVoice 10 (German version), was used to recover an original text from noisy signals. A microprocessor was placed between the speaker and the ViaVoice module and it performed an on-line high frequency noise suppression. The system intelligibility increases after application of our algorithm. We also present a detailed analysis of signal properties in the Fourier space that explains why the method works.

Although the experiment was successful, it is only the first step toward real applications. One should regard more complex methods of NR in order to remove noise without signal distortion at the same time. One should think also about a closer integration of the NR method with speech recognition programs in order to fully utilize the power of synergies. Further work should focus on a faster microprocessor with built-in mathematical functions that would make possible incorporating more complex algorithms. Our dynamic filtering, which was rather a simple code in the world of high level programming, used almost 90% of our microprocessor memory.

Acknowledgments

This paper, as a project of the ESF Programme "Stochastic Dynamics: fundamentals and applications" (STOCHDYN), was supported by the Polish Ministry of Science and Higher Education, grant No. ESF/275/2006. The authors are thankful to Prof. Hartmut Benner for illuminating discussions on noisy signals.

References

[1] K. Urbanowicz, H. Kantz, *Chaos* **17**, 023121 (2007).

[2] E.J. Kostelich, T. Schreiber, *Phys. Rev. E* **48**, 1752 (1993).

[3] T. Schreiber, *Phys. Rev. E* **48**, 13(4) (1993).

[4] T. Schreiber, *Phys. Rev. E* **47**, 2401 (1992).

[5] J.D. Farmer, J.J. Sidorowich, *Physica D* **47**, 373 (1991).

[6] S.M. Hammel, *Phys. Lett. A* **148**, 421 (1990).

[7] M.E. Davies, *Physica D* **79**, 174 (1994).

[8] R. Cawley, G.H. Hsu, *Phys. Rev. A* **46**, 3057 (1992).

[9] T. Sauer, *Physica D* **58**, 193 (1994).

[10] P. Grassberger, R. Hegger, H. Kantz, C. Schaffrath, T. Schreiber, *Chaos* **3**, 127 (1993).

[11] K. Urbanowicz, J.A. Hołyst, *Acta Phys. Pol. B* **36**, 2805 (2005), arXiv:cond-mat/0411324.

[12] K. Urbanowicz, J.A. Hołyst, T. Stemler, H. Benner, *Acta Phys. Pol. B* **35**, 2175 (2004); arXiv:cond--mat/0308554.

[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, in: *C. The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge 2002.

[14] R. Hegger, H. Kantz, L. Matassini, *Phys. Rev. Lett.* **84**, 3197 (2000).

[15] H. Kantz, R. Hegger, L. Matassini, *IEEE Trans. Circuits, Systems I* **48**, 1454 (2001).

[16] Linguatec web page: `http://www.linguatec.de` .